PEMBANGUNAN APLIKASI INFORMASI KESEHATAN MASYARAKAT KOTA MALANG BERBASIS *MOBILE NATIVE ANDROID*

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh:
Ferdy Wahyurianto
NIM: 135150200111011

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2018

PENGESAHAN

PEMBANGUNAN APLIKASI INFORMASI KESEHATAN MASYARAKAT KOTA MALANG
BERBASIS MOBILE NATIVE ANDROID

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Ferdy Wahyurianto
NIM: 135150200111011

Skripsi ini telah diuji dan dinyatakan lulus pada
2 Agustus 2018
Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Issa Arwani, S.Kom, M.Sc	Arief Andy Soebroto, S.T, M.Kom
NIP: 19830922 201212 1 003	NIP: 19720425 199903 1 002

Mengetahui
Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T, M.T, Ph.D
NIP: 19710518 200312 1 001

IDENTITAS PENGUJI

- Dosen Penguji I

  Dr. Eng.Herman Tolle, S.T, M.T

  NIK. 19740823 200012 1 001

- Dosen Penguji II

  Ir.Sutrisno, M.T

  NIK. 19570325 198701 1001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 2 Agustus 2018

Ferdy Wahyurianto

NIM: 135150200111011

DAFTAR RIWAYAT HIDUP

Nama	: Ferdy Wahyurianto
Tempat, Tanggal Lahir	: Sumenep, 23 Februari 1995
Alamat Asal	: Jl. Dorang no. 16 Desa Kolor, Sumenep.
Nama Orang Tua	: Irianto
Riwayat Pendidikan	: SD Pajagalan 1 Sumenep (2001-2007)
	SMP Negeri 1 Sumenep (2007-2010)
	SMA Negeri 1 Sumenep (2010-2013)
	S1 Informatika Universitas Brawijaya (2013-2018)
Alamat di Malang	: Jl. Candi Jago no.8 Kelurahan Blimbing, Malang
No. telpon/HP	: 0821662623
E-mail	: ferdywr@gmail.com
Prestasi	: -
Pengalaman Kepanitiaan	: -
Pengalaman Organisasi	: -

UCAPAN TERIMA KASIH

Ucapan terima kasih penulis haturkan kepada seluruh pihak yang selama ini telah mendukung dan membantu dalam proses penelitian skripsi. Kepada dosen pembimbing I dan dosen pembimbing II yang telah meluangkan waktunya dalam membimbing dan memberikan masukan berharga bagi skripsi ini. Ucapan terima kasih juga penulis haturkan kepada teman-teman di kampus yang telah memberikan dukungan dan berjuang bersama-sama dalam menyelesaikan penelitian. Semoga segala jerih payah perjuangan yang telah dilakukan bisa memberi manfaat bagi banyak orang.

Malang, 2 Agustus 2018

Penulis

ferdywr@gmail.com

ABSTRAK

Ferdy Wahyurianto, Pembangunan Aplikasi Informasi Kesehatan Masyarakat Kota Malang Berbasis *Mobile Native Android*

Dosen Pembimbing:

1. **Issa Arwani, S.Kom, M.Sc**
2. **Arief Andy Soebroto, S.T, M.Kom**

Populasi masyarakat akan terus berkembang seiring berjalannya waktu, hal ini membuat proses pendataan kesehatan masyarakat menjadi lebih sukar untuk dilakukan. Masalah tersebut menjadi alasan dalam mengembangkan teknologi untuk mempermudah proses pendataan kesehatan masyarakat secara cepat. Informasi kesehatan masyarakat tersebut digunakan oleh pemerintah dalam melayani dan meningkatkan kesejahteraan masyarakat. Teknologi informasi kesehatan masyarakat di Kota Malang sekarang masih berbasis website yang membutuhkan tempat yang statis dan mobilitas yang rendah. Penyelesaian untuk masalah tersebut, yaitu dengan mengembangkan aplikasi *mobile* yang lebih praktis dan banyak diminati, karena memberikan kemudahan untuk mengakses informasi dan mendukung mobilitas yang tinggi. Solusi yang ditawarkan penulis adalah membangun sebuah aplikasi informasi kesehatan masyarakat berbasis *mobile native* dengan menggunakan *platform android* karena mayoritas 89,75% masyarakat menggunakan sistem operasi *android*. Metode pengembangan aplikasi ini menggunakan metode waterfall yang mendukung pengembangan sesuai dengan kebutuhan pengguna yang didefinisikan secara jelas di awal. Aplikasi ini dapat mengolah dan memproyeksikan jumlah persebaran data kesehatan masyarakat kedalam *Google Maps*. Sistem ini diuji dengan melakukan *unit testing*, *validation testing*, serta *compatibility testing*. Hasil analisis pengujian pada sistem ini memenuhi kebutuhan pengguna serta mampu menunjang kinerja sistem secara cepat dan mobilitas tinggi.

Kata Kunci: *android*, *native*, kesehatan masyarakat, *waterfall*, *google maps*, *firebase*, *json*, *javascript*, *nodejs*, *compatibility*, *whitebox*, *blackbox*.

ABSTRACT

Ferdy Wahyurianto, ***Development of Society's Health Information Application of Malang Based on Mobile Native Android***

Supervising Lecturer:

1. **Issa Arwani, S.Kom, M.Sc**
2. **Arief Andy Soebroto, S.T, M.Kom**

The population of the society will continue to evolve over time, this makes process of logging society's health data will be more difficult to do. The problem is becoming a foundation in developing technology to ease the process of logging society's health quickly. The society's health information used by the Government in serving and improving the welfare of the society. The society's health information technology at Malang present day is based with website which requires a static place and low mobility. The solution to this problem is by developing a mobile app that is more practical and much sought present day, because it gives convenience to access information and high mobility support. Solutions offered the author is building a society's health information system based on mobile native app using the android platform because the majority of 89.75% of society use android operating system. System development method using the waterfall method that supports development in accordance with the needs of users who are clearly defined at the beginning. This Application can process and projecting the amount of society's health data distribution into Google Maps. The system is tested by doing unit testing, validation testing, as well as the compatibility testing. The results of the testing analysis on this system meets the needs of users as well as being able to support the system's performance and high mobility.

Keywords : android, native, society's health, waterfall, google maps, firebase, json, javascript, nodejs, compabitility, whitebox, blackbox.

KATA PENGANTAR

Dengan segala kerendahan hati dan diri memanjatkan syukur kehadirat Allah SWT atas limpahan rahmatNya sehingga penulis dapat menyelesaikan penulisan skripsi dengan baik. Penulisan skripsi diajukan untuk memenuhi salah satu persayaratan dalam memperoleh gelar Sarjana Komputer pada Fakultas Ilmu Komputer Universitas Brawijaya. Topik skripsi yang diajukan adalah "Pembangunan Aplikasi Informasi Kesehatan Masyarakat Kota Malang Berbasis *Mobile Native Android*". Kelancaran penulisan skripsi ini tidak lepas dari bantuan berbagai pihak, oleh karena itu penulis mengucapkan terimakasih dan rasa hormat kepada:

1. Bapak Agus Wahyu Widodo, S.T, M.Cs selaku ketua Program Studi Teknik Informatika Fakultas Ilmu Komputer Universitas Brawijaya.
2. Bapak Issa Arwani, S.Kom, M.Sc selaku dosen pembimbing I yang telah memberikan arahan dan menyediakan waktu untuk berdiskusi dengan penulis selama proses penulisan skripsi.
3. Bapak Arief Andy Soebroto, S.T, M.Kom selaku dosen pembimbing II yang telah memberikan banyak masukan dan motivasi kepada penulis selama proses penulisan skripsi.
4. Direktur CV. Sarana Utama Solusindo yang telah memberikan kesempatan kepada penulis untuk melakukan penelitian.
5. Bapak Irianto dan Ibu Susiantika Wiandari selaku orang tua serta Desy Ayurianti selaku adik yang selalu memberikan doa dan segala macam bentuk dukungan yang selalu diberikan kepada penulis.
6. Seluruh rekan team KRS yang telah telah memberikan banyak bantuan dan semangat selama pengerjaan skripsi.
7. Ardana Prakasita Devi selaku teman dekat yang telah membantu dan membimbing dalam penulisan skripsi.
8. Seluruh rekan Museum Kopi yang telah memberikan fasilitas dan dukungan dalam pengerjaan skripsi.
9. Seluruh anggota YVCI Sumenep Chapter yang telah memberikan semangat dalam pengerjaan skripsi.
10. Rekan-rekan seperjuangan skripsi yang telah banyak mambantu saat proses pengerjaan skripsi.
11. Seluruh civitas academica Fakultas Ilmu Komputer Universitas Brawijaya.

Penulis menyadari bahwa dalam penulisan skripsi ini masih jauh dari kata sempurna, oleh karenanya segala kritik dan saran yang bersifat membangun sangat penulis harapkan demi perbaikan skripsi ini. Semoga karya sederhana ini dapat memberikan manfaat bagi semua pihak sekaligus sebagai sarana pendukung pada pengembangan karya-karya terkait dimasa mendatang.

Malang, 20 Juli 2018

Ferdy Wahyurianto
ferdywr@gmail.com

DAFTAR ISI

DAFTAR TABEL

BAB 1 PENDAHULUAN

Pada bab ini membahas tentang latar belakang masalah yang diangkat menjadi topik permasalahan, rumusan masalah, tujuan penelitian, manfaat penelitian kepada penulis maupun terhadap masyarakat, batasan masalah serta sistematika pembahasan pada penelitian ini.

1.1 Latar belakang

Perkembangan teknologi yang pesat pada abad ini, mendorong masyarakat untuk bersikap lebih modern dan dituntut untuk dapat mengikuti perkembangan teknologi yang berkembang dengan cepat. Berbagai proses sederhana yang dilakukan pada pelayanan masyarakat, selayaknya harus diperbaharui untuk lebih sesuai dengan perkembangan teknologi dalam proses pelayanan masyarakat. Pembaharuan teknologi merupakan kewajiban yang harus dilaksanakan dalam mengatasi masalah yang terjadi dari bertambahnya populasi masyarakat yang semakin banyak. Bertambahnya populasi masyarakat disini dapat menimbulkan berbagai masalah pada setiap kalangan masyarakat. Berbagai aspek masalah yang terjadi pada populasi masyarakat ini membuat proses sederhana menjadi lebih sulit untuk dilakukan. Proses sederhana dalam pelayanan kesehatan membutuhkan waktu, proses, dan pengolahan data yang panjang pada setiap individu masyarakat dan akan menimbulkan masalah lainnya. Kelola akses data pada proses sederhana dapat digantikan dengan suatu sistem informasi yang lebih cepat, mudah serta lebih akurat (Susilowati & Riasti, 2011).

Pelayanan kesehatan masyarakat merupakan salah satu bidang pengembangan yang menggunakan sistem informasi untuk kelola akses data kesehatan masyarakat. Pelayanan kesehatan harus dikembangkan dan diupayakan oleh setiap masyarakat dan pemerintah pada setiap daerah agar proses pelayanan lebih cepat, mudah serta lebih akurat (UU, 2009). Aplikasi *mobile* merupakan salah satu teknologi yang berpotensi, dan dapat digunakan dalam pengumpulan, penyimpanan, serta pengolahan data pada setiap individu dalam populasi masyarakat (Safaat H, 2012). Kemudahan dan kemobilitasan yang ditawarkan oleh aplikasi pada perangkat mobile menjadi tujuan utama mengapa sistem informasi ini dapat digunakan dan dikembangkan melalui perangkat *mobile*.

Teknologi sistem informasi yang dibutuhkan dalam pelayanan kesehatan masyarakat adalah sistem yang dapat melakukan pengaksesan, penyimpanan dan pengolahan data. Tata kelola data tersebut tentunya harus dapat dilakukan dalam satu lingkup sistem informasi yang sama dan membentuk satu kesatuan komponen pada sistem informasi (Wardiana, 2002). Pada penjelasan kebutuhan diatas menjadi alasan digunakannya aplikasi informasi yang dapat digunakan pada perangkat *mobile* yang praktis, dengan hanya menggunakan *smartphone* dan koneksi internet. Kebutuhan-kebutuhan diatas menjadi dasar dalam memenuhi pelayanan kesehatan masyarakat dengan membangun sebuah aplikasi berbasis *mobile* yaitu Aplikasi Informasi Kesehatan Masyarakat.

Aplikasi ini berbasis *mobile* dan dibangun dengan *platform android* karena mayoritas penduduk Indonesia menggunakan *platform android* sebanyak 89,75% (StatCounter, 2017). Pengembangan pada sistem ini menggunakan pendekatan secara *native*. Pengembangan aplikasi *mobile native* dikembangkan secara khusus untuk digunakan pada satu *platform* saja tidak berlaku untuk *platform* lainnya. Pengembangan secara *native* juga dapat menggunakan hak akses pada seluruh *hardware* yang ada pada perangkat *mobile,* seperti *Camera, Global Positioning System (GPS),* dan sebagainya (IBM Corporation, 2012). *Android native* memiliki keunggulan dalam grafik *user interface (UI)* serta *user experience (UX)* terbaik dibandingkan dengan pendekatan yang lain. Selain itu pada aplikasi ini membutuhkan kemanan, performa dan pemeliharaan sistem yang tinggi yang dimana hal itu merupakan keunggulan dari aplikasi *native* (Kumar S, 2014). Perangkat *mobile* memiliki keunggulan untuk dapat bekerja saat *offline* atau tanpa koneksi internet (Michael, et al., n.d.). *Mobile native* menggunakan bahasa *java* sebagai bahasa pemrogramannya, dan menggunakan bahasa *Extensible Markup Language* (*XML*) untuk tampilan *layout* dan desain sistem (Google Inc. & Android Developers, 2017). Pengembangan secara *native* disini dipilih karena pada memiliki kecepatan yang lebih cepat, serta performa yang lebih baik dari pada pengembangan secara *hybrid* (Optimus Information, 2015).

Pada pelayanan kesehatan masyarakat dibutuhkan sebuah pelayanan yang memberikan informasi secara *real time monitoring* yang memudahkan dalam pelayanan masyarakat. Pelayanan dan tindakan yang efektif mempengaruhi kenyamanan dan kondisi pasien. Semakin besar ruang lingkup jasa pelayanan maka akan semakin kompleks jenis tindakan dan pelayanan yang harus diberikan, maka dari itu harus tetap dalam satu integrasi. Dalam rangka meningkatkan pelayanan kesehatan masyarakat Kota Malang pada pengolahan data dan informasi kesehatan pada ibu hamil, penderita gizi buruk, dan penderita *tubercolosis.* Oleh karena hal tersebut dibutuhkan sebuah sistem yang dapat memenuhi kebutuhan pelayanan kesehatan masyarakat yang memiliki urgensi tinggi.

Dukungan sistem dalam pelayanan informasi kesehatan di Kota malang saat ini masih berbasis website yang kurang memiliki aspek yang memadai dalam penanganan cepat. Penggunaan perangkat komputer atau *desktop* masih kurang efektif dalam pelayanan kesehatan secara cepat, oleh karena itu maka dilakukan penelitian agar sistem informasi tersebut dapat bekerja lebih optimal. Aplikasi *mobile native* menjadi sebuah pilihan karena mudah dan lebih praktis dalam akses informasi secara *mobile*, serta memiliki performa yang stabil. Pada aplikasi ini terdapat fitur dalam memproyeksikan jumlah persebaran data kesehatan masyarakat kedalam *Google Maps* yang dapat mengetahui tingkat persebaran setiap daerah yang meliputi 3 informasi kesehatan pada Kota Malang. Hal ini dilakukan untuk mengetahui tingkat tertinggi hingga terendah pada daerah yang membutuhkan penanganan kesehatan secara cepat. Penyaringan dan pencarian informasi juga akan dilakukan pada aplikasi ini agar dapat membantu proses kelola akses data kesehatan masyarakat dengan cepat, akurat, serta lebih optimal.

Aplikasi pada *mobile native* juga lebih mudah untuk digunakan dimana saja tanpa harus membutuhkan perangkat komputer *desktop,* sehingga akses informasi kesehatan dapat dilakukan dimana saja. Maka dengan permasalahan tersebut, dibuatlah penelitian dengan judul "Pembangunan Aplikasi Informasi Kesehatan Masyarakat Kota Malang berbasis *Mobile Native Android"* dengan nama SIMKES.

1.2 Rumusan masalah

Berdasarkan uraian latar belakang di atas, maka dapat dirumuskan permasalahan sebagai berikut:

1. Bagaimana hasil analisis kebutuhan pada aplikasi informasi kesehatan masyarakat Kota Malang berbasis *Mobile Native Andorid* dengan metode *waterfall?*
2. Bagaimana hasil perancangan pada aplikasi informasi kesehatan masyarakat Kota Malang berbasis *Mobile Native Andorid* dengan metode *waterfall?*
3. Bagaimana hasil implementasi pada aplikasi informasi kesehatan masyarakat Kota Malang berbasis *Mobile Native Andorid* dengan metode *waterfall?*
4. Bagaimana hasil pengujian pada aplikasi informasi kesehatan masyarakat Kota Malang berbasis *Mobile Native Andorid* dengan metode *waterfall?*
5. Bagaimana menerapkan dan menampilkan informasi kesehatan masyarakat kedalam *Google Maps?*

1.3 Tujuan

Tujuan penelitian ini adalah :

1. Mengembangkan aplikasi mobile native yang dapat mengolah data menjadi sebuah informasi dalam aplikasi informasi kesehatan masyarakat yang sesuai dengan kebutuhan pengguna.
2. Mengembangkan aplikasi yang dapat membantu mempermudah pengolahan dan pemetaan data kesehatan masyarakat Kota Malang.

1.4 Manfaat

Manfaat yang terdapat dalam skripsi ini adalah:

a. Bagi pemerintah:

1. Menghasilkan aplikasi *mobile native* informasi kesehatan masyarakat yang dapat membantu pendataan masalah kesehatan secara cepat dant tepat.
2. Menyediakan informasi tetang persebaran masalah kesehatan di daerah sekitar malang.
3. Memberikan kemudahan dalam pengambilan data masyarakat untuk pelayanan kesehatan masyarakat Kota Malang.

b. Bagi penulis :

1. Membangun aplikasi *mobile native* yang dapat membantu proses pelayanan kesehatan untuk masyarakat.
2. Mengungkapkan pemikiran dan hasil penelitian dalam bentuk tulisan yang sistematis dan metodologis.
3. Mampu menghasilkan pemikiran dan penelitian dalam bentuk karya ilmiah.

1.5 Batasan masalah

Batasan masalah yang terdapat dalam skripsi ini adalah sebagai berikut :

1. Proses pembangunan aplikasi informasi kesehatan masyarakat menggunakan *mobile native*.
2. Pembangunan aplikasi informasi kesehatan masyarakat menggunakan *Android Studio* dan menggunakan bahasa pemrograman *Java.*
3. Data yang digunakan adalah data kesehatan yang meliputi ibu hamil, balita, gizi buruk dan penyakit *Tubercolosis* yang diambil dari Dinas Kesehatan Kota Malang.
4. Aplikasi informasi kesehatan masyarakat Kota Malang berbasis *Mobile Native Andorid* menggunakan *google firebase* sebagai simulasi pengembangan aplikasi informasi kesehatan Kota Malang.

1.6 Sistematika pembahasan

Sistematika pembahasan yang terdapat dari pembangunan aplikasi *mobile native* ini diuraikan sebagai berikut:

BAB 1 Pendahuluan

Bab ini menjelaskan latar belakang serta tujuan dilakukannya penelitian tentang "Pembangunan Aplikasi Informasi Kesehatan Masyarakat Kota Malang berbasis *Mobile Native Android*" termasuk juga perumusan masalah, manfaat penelitian, batasan penelitian, sistematika pembahasan, dan jadwal penelitian.

BAB 2 Landasan Kepustakaan

Bab ini menjelaskan tentang teori-teori pendukung dan kajian pustaka yang terkait dengan pembangunan yang telah ada seperti pembangunan aplikasi mobile dengan menggunakan pendekatan secara native menggunakan *android studio*, bahasa pemrograman *Java*, *Extensible Markup Language* (*XML*) dan lainnya.

BAB 3 Metodologi

Bab ini menjelaskan metode-metode penelitian yang digunakan seperti metode perancangan, metode pengujian, dan metode lain yang berhubungan dengan aplikasi mobile native pada sistem pelayanan kesehatan masyarakat.

BAB 4 Analisis Kebutuhan

Bab ini menjelaskan analisis kebutuhan aplikasi *mobile native* pada sistem pelayanan kesehatan masyarakat. Analisis kebutuhan direpresentasikan

dengan menerapkan gambaran umum sistem, identifikasi aktor, kebutuhan fungsional maupun non-fungsional, *usecase diagram,* dan *usecase scenario*.

BAB 5 Perancangan

Bab ini menjelaskan perancangan aplikasi berdasarkan analisis kebutuhan, dan akan dirancang menggunakan arsitektur sistem, perancangan *sequence* diagram, *class* diagram, *activity* diagram dan perancangan *interface*.

BAB 6 Implementasi

Bab ini menjelaskan tentang implementasi aplikasi mobile native yang berdasarkan pada metode-metode yang akan digunakan, serta prosedur-prosedur yan akan dilakukan pada implementasi aplikasi *mobile native*.

BAB 7 Pengujian dan Analisis

Bab ini menjelaskan tentang hasil pengujian dari pembangunan aplikasi informasi kesehatan masyarakat dan hasil analisis dari proses pembangunan aplikasi informasi kesehatan masyarakat.

BAB 8 Penutup

Bab ini menjelaskan tentang kesimpulan dan saran dari pelaksanaan Pembangunan Aplikasi Informasi Kesehatan Masyarakat Kota Malang berbasis *Mobile Native Android*.

BAB 2 LANDASAN KEPUSTAKAAN

Pada bab ini menjelaskan teori-teori, konsep, model, metode dari literatur yang berkaitan dalam pembangunan aplikasi yang digunakan dalam penelitian skripsi ini. Kajian pustaka akan menjelaskan penelitian dan teori dari berbagai sumber pustaka yang pernah diterapkan pada penelitian sebelumnya dan yang berkaitan dengan penyusunan skripsi serta kebutuhan dalam penelitian ini. Dasar teori adalah penjelasan tentang teori yang diperlukan dalam perancangan aplikasi.

2.1 Kajian Pustaka

Kajian pustaka berisi tentang uraian penelitian yang sebelumnya pernah dilakukan yang berkaitan dengan Aplikasi Informasi Kesehatan, penggunaan *mobile native android* dan penggunaan *google firebase* sebagai simulasi untuk proses pertukaran data. Penelitian ini dilakukan dengan menggabungkan metode, objek, dan kebutuhan yang digunakan pada penelitian sebelumnya.

Penelitian tentang "Implementasi Aplikasi Sistem Informasi Kesehatan Daerah (SIKDA) Generik Di UPT. Puskesmas Gambut Kabupaten Banjar" yang dilakukan oleh Khairina Isnawati, Eko Nugroho, dan Lutfan Lazuardi pada tahun 2016. Penelitian ini dijelaskan mengenai penerapan tentang teknologi sistem informasi kesehatan daerah pada UPT. Puskesmas Gambut Kabupaten Banjar yang dilakukan untuk mempermudah proses pelaporan kesehatan yang banyak dengan menggunakan metode penelitian deskriptif. Data diperoleh dengan metode wawancara pada penelitian deskriptif ini. Hasil dari penelitian menunjukkan bahwa kemampuan perangkat lunak harus diperhatikan dalam implementasi sistem informasi kesehatan daerah (Isnawati, et al., 2016).

Pada penelitian tentang "Perancangan Sistem Informasi Kesehatan (Puskesmas Keliling) Berbasis Web" yang dilakukan oleh Rachmat Agusli, Lilis Sakuroh, dan Nopriyadi pada tahun 2016. Penelitian ini menjelaskan tentang implementasi teknologi pencatatan dan kelola data pasien puskesmas keliling di puskesmas kecamatan Pademangan yang berbasiskan *web.* Metode yang digunakan dalam perancangan menggunakan *Unified Modelling Language (UML)* dengan desain tampilan menggunakan aplikasi *Dreamweaver* dan proses *coding* menggunakan bahasa pemrograman *PHP* dan *MySQL*. Penelitian ini menghasilkan teknologi yang digunakan untuk mengganti proses pencatatan dan kelola data yang dilakukan secara manual. Kelola data secara manual tersebut diubah ke dalam sistem yang terkomputerisasi yang mendukung proses kelola kebutuhan informasi yang cepat , akurat, dan tepat waktu (Agusli, et al., 2016).

Pada penelitian tentang "Gambaran Penerapan Sistem Informasi Manajemen Kesehatan Berbasis Web di Puskesmas Kota Makassar" yang dilakukan oleh Dwi Santy Damayati, Muhammad Rusmin, dan Zilfadhilah Arranury pada tahun 2015. Penelitian ini menjelaskan tentang penerapan e-puskesmas yang berbasis web untuk meningkatkan kinerja management kesehatan. Metode penelitian ini menggunakan metode kuantitatif dengan metode penelitian deskriptif serta perolehan data dari metode *total sampling*. Hasil dari penelitian ini membuktikan bahwa pada sisi *hardware* dan *network* kurang mendukung serta

data dan pelatihan dari setiap operator kurang memadai. Hal ini membuktikan bahwa pada pelayanan kesehatan membutuhkan teknologi yang lebih mutakhir dari penggunaan teknologi *web* (Damayanti, et al., 2015).

Pada penelitian tentang "Sistem Informasi Publik Layanan Kesehatan menggunakan Metode Location Based Service di Kota Semarang" yang dilakukan oleh Jefri Alfa Razaq dan Arief Jananto pada tahun 2014. Penelitian ini menjelaskan tentang penerapan sistem informasi lokasi penting pelayanan kesehatan untuk masyarakat kota Semarang. Model pengembangan pada penelitian ini menggunakan model *System Development Life Cycle (SDLC).* Analisa kebutuhan dimodelkan degan *usecase diagram, activity diagram, class diagram, database,* dan *E-R diagram.* Penelitian ini menghasilkan suatu aplikasi pelayanan kesehatan masyarakat yang memiliki fitur pemetaan lokasi geografis pelayanan kesehatan terdekat beserta rute yang dilalui (Razaq & Junanto, 2014).

Pada penelitian "Desain UML Aplikasi Navigasi Layanan Kesehatan Berbasis Android" yang dilakukan oleh Sariyun Naja Anwar, Fatkhul Amin, dan Isworo Nugroho pada tahun 2014. Penelitian ini menjelaskan tentang pemanfaatan potensi *smartphone* untuk penyebaran informasi dan *navigasi* lokasi pelayanan kesehatan. Pemodelan sistem dilakukan dengan menggunakan *usecase diagram, Activity diagram, sequence diagram,* dan *class diagram.* Penelitian ini menghasilkan aplikasi penyedia informasi lokasi sebagai *navigasi* pada pelayanan kesehatan masyarakat kota Semarang (Anwar, et al., 2014).

Pada penelitian "Choosing a Mobile Application Development Approach" yang dilakukan oleh Phyo Min Tun yang berasal dari *Stanford International University*, Thailand pada tahun 2014. Penelitian ini menjelaskan tentang memilih pendekatan pada pengembangan aplikasi *mobile* yang sesuai dengan kebutuhan *developer* atau pengembang sistem dengan membandingkan setiap pendekatan. Terdapat tiga pendekatan pada pengembangan aplikasi *mobile* yaitu pendekatan *native, hybrid,* atau aplikasi *mobile web* yang dapat digunakan pada keputusan untuk pengembangan aplikasi *mobile.* Perbandingan pada setiap pendekatan dilakukan dengan metode diskusi dan analisa pada kelebihan dan kekurangan pada setiap pendekatan. Penelitian ini menghasilkan kesimpulan yaitu pada pendekatan native, lebih cocok untuk aplikasi yang membutuhkan performa yang cepat, namun akan membutuhkan biaya yang tidak sedikit. Pada pendekatan secara *mobile web* lebih cocok untuk aplikasi yang membutuhkan sedikit biaya, namun terkendala pada penggunaan fungsi dan performa. Pada pendekatan secara hybrid lebih cocok untuk aplikasi yang digunakan untuk *multi platform* dengan perbandingan performa dan biaya yang cukup (Tun, 2014).

Pada penelitian "Rancang Bangun Aplikasi Mobile untuk Notifikasi Jadwal Kuliah Berbasis Android" pada studi kasus STMIK Provisi Semarang yang dilakukan oleh Taufik Ramadhan dan Victor G Utomo pada tahun 2014. Penelitian ini menjelaskan tentang penerapan aplikasi mobile native untuk notifikasi jadwal kuliah untuk mahasiswa STMIK Provisi Semarang untuk perbaikan kualitas pelayanan jadwal kuliah. Perbaikan tersebut dikhususkan untuk notifikasi informasi jadwal pada setiap perangkat *mobile* setiap mahasiswa. Notifikasi penjadwalan tersebut dikembangkan menggunakan metode *System Development*

Life Cycle (SDLC). Pendekatan menggunakan *waterfall* dalam *SDLC* dipilih dalam proses pengembangan sistem, analisa kebutuhan dimodelkan degan *usecase diagram.* Penelitian ini menghasilkan aplikasi *mobile* yang dapat menjadi pilihan dalam akses informasi jadwal kuliah yang dapat mempermudah mahasiswa dalam mendapatkan informasi jadwal perkuliahan dengan fitur notifikasi (Ramadhan & Utomo, 2014).

Pada penelitian "Pembangunan Aplikasi Brawijaya Messenger dengan menggunakan *Platform Google Firebase* pada Universitas Brawijaya" yang dilakukan oleh Afifur Rozaq pada tahun 2018. Penelitian ini dilakukan untuk mempermudah komunikasi antar mahasiswa Universitas Brawijaya yang menggunakan *smartphone* untuk saling bertukar informasi. Selain itu penelitian ini bertujuan untuk menunjang kegiatan akademik mahasiswa melalui group chat yang berisikan seluruh anggota kelas termasuk dosen. Dengan adanya group chat tersebut informasi yang akan disampaikan akan diketahui oleh semua anggota group kelas. Penelitian ini menggunakan *google firebase* dalam media penyimpanan informasi secara *realtime.* Metodologi yang digunakan dalam penelitian adalah menggunakan metode *System Development Life Cycle (SDLC) dengan pendekatan waterfall.* Pertukaran informasi dan data secara *online* menggunakan respon data *JSON* (Rozaq, et al., 2018).

Penelitian yang dilakukan oleh Argo Wibowo dengan judul "Perancangan Aplikasi Konsultasi Ibu Hamil Berbasis *Cloud Computing*" dilakukan untuk menghubungkan antara ibu hamil dengan dokter. Kebenaran informasi dalam konsultasi ibu hamil dengan dokter dapat divalidasi dengan mudah dan cepat, selain itu penelitian ini membantu ibu hamil untuk saling bercengkrama tentang pengalamannya dengan dokter kandungan untuk kebaikan kandungan. Penelitian ini berbasiskan *cloud computing* dengan menggunakan metode penelitian *prototyping apps* dikarenakan pengguna tidak seluruhnya paham kebutuhan apa yang akan dijadikan fitur pada aplikasi ini. *Google firebase* menjadi media penyimpanan dan pertukaran informasi maupun data *online* secara *realtime* pada *cloud computing* aplikasi ini (Wibowo, 2018).

Pada penelitian "Pengembangan Sistem informasi Kependudukan Berbasis Mobile dan *RESTful Web Service*" yang dilakukan oleh Rachel Kurniawati pada tahun 2016. Penelitian ini menjelaskan tentang penerapan aplikasi *mobile* dalam memperbaiki metode pengambilan data sensus penduduk yang dilakukan secara *door-to-door* yang kurang efektif. Integrasi *RESTful Web Service* dengan aplikasi android menjadi pemecahan dalam masalah efektifitas pengambilan data sensus penduduk agar lebih sistematik dan efisien. Metode yang digunakan dalam perancangan aplikasi menggunakan metode *System Development Life Cycle (SDLC)* dengan pendekatan *waterfall.* Penelitian ini menghasilkan aplikasi *mobile* yang dapat membantu proses pengambilan data sensus penduduk secara efektif dan efisien (Kurniawati, 2016).

Berdasarkan beberapa penelitian diatas, penulis mengusulkan penelitian yang berjudul "Pembangunan Aplikasi Informasi Kesehatan Masyarakat Kota Malang berbasis *Mobile Native Android*". Penelitian ini menggunakan metode *System Development Life Cycle (SDLC)* dengan pendekatan *Waterfall.* Analisis

kebutuhan dilakukan dengan metode wawancara. Hasil dari analisis kebutuhan yang didapatkan akan direpresentasikan dengan *Usecase diagram.* Penelitian ini dapat mempermudah dalam pengambilan data masyarakat untuk pelayanan kesehatan masyarakat Kota Malang dan dapat membantu pemerintah dalam melakukan pelayanan kesehatan terhadap masyarakat.

2.2 Pembangunan

Pembangunan aplikasi untuk mendukung dan mewujudkan kesehatan masyarakat merupakan usaha untuk meningkatkan derajat kesehatan masyarakat yang dilandasi dengan wawasan kesehatan sebagai bentuk pembangunan nasional (UU, 2009). Pada sisi teknologi, pembangunan merupakan bagian awal yang dikerjakan setelah pengembangan dan analisa aplikasi dalam mendefinisikan kebutuhan pendukung kinerja fungsi dari aplikasi yang akan dibangun. Dalam tahap ini visualisasi awal aplikasi sangat dibutuhkan agar sesuai dengan tujuan pembuatan aplikasi itu sendiri. Analisis perencanaan, *sketching*, dan pengaturan kebutuhan sehingga menjadi beberapa komponen yang terpisah yang kemudian digabungkan. Penggabungan komponen ini dilakukan hingga menjadi satu komponen yang terintegrasi dan berfungsi secara fungsional sesuai dengan apa yang direncanakan di awal.

Pengaturan dari keseluruhan komponen juga termasuk dalam fase rancang bangun ini. Komponen disini adalah komponen pembangun aplikasi seperti *Software, Hardware,* dan *Brainware*. Pengaturan disini ditujukan untuk singkronisasi atau penyelarasan hubungan diantara kerja *Software* dengan *Hardware*, *Software* dengan *Brainware* serta *Hardware* dengan *Brainware*. Agar dalam setiap bagian bekerja secara optimal.

2.3 Sistem Informasi Kesehatan Masyarakat

Sistem informasi merupakan pengelompokan komponen-komponen seperti manusia, proses, atau data sebagai objek untuk menjadi sebuah informasi dari hasil penyimpanan, pemrosesan, dan pengkolektifan pada hasil akhirnya. Informasi inilah yang dikelola, diolah untuk saling berinteraksi sebagai penyimpan, dan penyedia informasi yang digunakan untuk membantu pengembangan sebuah organisasi agar lebih baik. Selain itu Sistem informasi juga dapat menjadi objek untuk menganalisa sebuah permasalahan dengan menjalankan fungsi tertentu untuk penyelesaian dan penyebaran hasil dari analisa masalah dalam bentuk informasi yang sah (Whitten & Bentley, 2007).

Sistem Informasi dalam organisasi terdiri dari 5 komponen utama yang membangun Sistem Informasi itu sendiri, 5 komponen tersebut terdiri dari :

1. Perangkat keras (*Hardware*)

   Hardware disini bekerja sebagai komponen yang mendukung kegiatan pengolahan data masukan, proses, dan data keluar.

2. Perangkat lunak (*Software*)

Software disini berperan dalam proses, proses yang dikerjakan oleh suatu program komputer dengan fungsi-fungsi tertentu yang digunakan untuk mengolah data mentah menjadi data keluaran.

3. Database

   Database ini merupakan suatu kelompok data yang terintegrasi atau saling berkaitan dan terhubung dengan data lainnya yang dapat diolah dengan menggunakan *Software* dan disimpan dalam suatu *Hardware.*

4. Jaringan (*Network*)

   Network adalah suatu jaringan yang menghubungkan sistem *Software* secara *real time* dengan pengguna sistem yang bertujuan untuk menghubungkan keduanya dalam suatu jaringan kerja.

5. Manusia (*Brainware*)

   Brainware berperan penting didalam keseluruhan proses, dimana manusialah yang menjadi *manager, system analyst, programmer,* dan *operator* yang bertanggung jawab atas kinerja dan perawatan dalam sistem.

Kesehatan merupakan keadaan sehat secara fisik, mental, spiritual maupun sosial yang memungkinkan setiap orang dan lapisan masyarakat untuk dapat hidup secara sosial dan ekonomi. Kesehatan adalah sebuah hak untuk setiap manusia dan salah satu unsur kesejahteraan yang harus diwujudkan, karena merupakan cita-cita Indonesia. Setiap kegiatan yang merupakan upaya untuk menjaga dan memelihara serta meningkatkan kualitas kesehatan masyarakat adalah tanggung jawab seluruh pihak dalam lapisan masyarakat maupun pemerintah. Pada upaya yang dilakukan membutuhkan sumber daya yang merupakan segala bentuk dana, tenaga, pemikiran serta teknologi yang dilakukan pada semua pihak. Teknologi yang dibutuhkan adalah segala bentuk metode atau alat yang dapat membantu mencegah, mendiagnosa, dan menangani setiap permasalahan kesehatan masyarakat (UU, 2009).

Sistem informasi kesehatan merupakan suatu pengelolaan informasi di seluruh tingkat pemerintah secara sistematis dalam rangka penyelenggaraan pelayanan kepada masyarakat. Pada teknologi sistem informasi kesehatan yang sebelumnya telah dimplementasikan di Kota Malang berbentuk *website* dengan nama SIMKES. SIMKES adalah aplikasi sistem informasi kesehatan yang berlaku secara nasional yang menghubungkan secara online dan terintegrasi seluruh Puskesmas, Dinas Kesehatan Kota, Dinas Kesehatan Provinsi, dan Kementerian Kesehatan. Pengembangan secara bertahap dan berkesinambungan dilakukan dalam rangka meningkatkan pelayanan kesehatan dibidang fasilitas pelayanan kesehatan serta meningkatkan ketersediaan dan kualitas data dan informasi manajemen kesehatan melalui pemanfaatan teknologi informasi. Secara bertahap sistem ini akan dikembangkan sesuai kondisi dan kesiapan implementasi dari tingkat operasional (Dinas Kesehatan Kota Malang, 2017).

2.4 Unified Modelling Language (UML)

Unified Modelling Language merupakan notasi grafis yang digunakan untuk pemetaan atau mendeskripsikan perancangan perangkat lunak yang berorientasi objek. Bahasa pemodelan grafis perangkat lunak merupakan bahasa yang lebih sukar dari pada bahasa pemrograman untuk dirancang dalam pembangunan sistem. Oleh karena itu dalam pembangunan sistem pemodelan sangat dibutuhkan dan dilakukan sebelum melakukan penyusunan kode program (Fowler, et al., n.d.).

2.4.1 *Class Diagram*

Class diagram menggambarkan jenis-jenis objek dalam sistem beserta hubungan atau relasi antar *class*. *Class Diagram* juga menunjukkan sifat dan operasi *class* serta cara setiap *class* terhubung (Fowler, et al., n.d.).

2.4.2 *Activity Diagram*

Activity diagram adalah teknik dalam menggambarkan dan mendeskripsikan logika dan aktifitas secara beruntun dalam suatu prosedur proses bisnis serta alur kerja sistem. *Activity diagram* lebih sesuai digunakan dalam menggambarkan alur kerja paralel dalam suatu *activity* dalam sistem agar logika dalam suatu *activity* dapat terbaca dengan jelas (Fowler, et al., n.d.).

2.4.3 *Sequence Diagram*

Sequence Diagram adalah teknik menggambarkan urutan dari perilaku dalam suatu skenario dalam suatu proses, perilaku pada setiap objek yang terdapat pada skenario suatu proses. *Sequence diagram* juga menunjukkan jumlah objek dan pesan yang diterima antara objek-objek tersebut. S*equence diagram* digunakan pada proses penentuan perilaku beberapa objek pada suatu *usecase* (Fowler, et al., n.d.).

2.5 Aplikasi *Mobile*

Aplikasi *mobile* merupakan perangkat lunak yang memiliki desain untuk dapat mendukung mobilitas yang tinggi dan berjalan pada perangkat bergerak (*mobile device*) seperti *smartphone* dan computer tablet. Pada sisi pembangunan aplikasi *mobile* mempunyai beberapa pendekatan, diantaranya aplikasi *mobile native*, *mobile web*, *mobile hybrid* (Optimus Information, 2015).

2.5.1 Aplikasi *Mobile Native*

Aplikasi *mobile* dengan pendekatan secara *native* merupakan aplikasi yang dikembangkan secara spesifik dan hanya untuk satu *platform mobile* saja, contohnya pada penggunaan bahasa pemrograman *java* untuk aplikasi ber-*platform android* atau bahasa pemrograman *Objective-C/Swift* untuk aplikasi ber-*platform iOS.* Perbedaan ini memerlukan tim pengembang yang berbeda untuk setiap *platform* yang memiliki pengetahuan yang khusus untuk setiap *platform.* Kekurangan pengembangan aplikasi *mobile* dengan pendekatan *native* adalah biaya pengembangan yang relatif mahal (IBM Corporation, 2012).

Keuntungan yang didapatkan dari pengembangan aplikasi *mobile native* adalah salah satunya karena aplikasi dengan pendekatan *native* dikembangkan dengan mengikuti pengembangan yang telah tersedia dalam *environment* pada pengembangan *platform* tersebut, sehingga aplikasi memiliki performa yang cepat dan *interface* yang konsisten dengan *platform* yang digunakan oleh pengguna. Aplikasi *mobile native* juga memiliki akses untuk penggunaan sensor pada perangkat keras yang terdapat pada perangkat *mobile* seperti kamera, mikrofon,dan lain sebagainya. Selain itu pada aplikasi *mobile native* mendukung penggunaan *offline mode* untuk pengaksesan aplikasi (IBM Corporation, 2012).

Feature	***Native App***
Developmet language	*Native only*
Code portability and optimization	*None*
Access device-specific features	*High*
Leverage exixting knowledge	*Low*
Advance graphics	*High*
Upgrade flexibility	*Low (Always by way of app stores)*
Installation experience	*High (From app store)*

Gambar 2.1 Fitur-fitur *Native Apps*

2.5.2 Aplikasi *Mobile Web*

Aplikasi *mobile web* merupakan website biasa seperti pada umumnya, namun *mobile-friendly* atau dapat digunakan dan dapat menyesuaikan dengan perangkat *mobile*. Pengembangan aplikasi *mobile web* ini menggunakan teknologi *web CSS, HTML5,* serta *javascript*. Pengguna dapat mengakses dengan mengetikkan alamat *URL* pada *browser* dan dapat mengunduh aplikasi dengan memberi *bookmark* halaman *mobile web* dan kemudian diakses melalui *homescreen* (IBM Corporation, 2012). Pengembangan *mobile web* cukup mudah, hanya dengan pengetahuan pengembangan *web* yang disesuaikan dalam penggunaan pada perangkat *mobile. Mobile web* ini tidak terikat pada *platform* tertentu, namun lebih kepada *browser* yang digunakan oleh pengguna. Karena itu, pengujian bukan dilakukan berdasar *platform,* namun akan dilakukan pada setiap *browser* yang berbeda. Tipe aplikasi *mobile web* juga dapat dikatakan sebagai aplikasi *cross-platform* karena tidak memperdulikan penggunaan *platform* yang berbeda.

Feature	***Native App***
Developmet language	*Web only*
Code portability and optimization	*High*
Access device-specific features	*Low*
Leverage exixting knowledge	*High*
Advance graphics	*Medium*
Upgrade flexibility	*High*
Installation experience	*Medium (By way of mobile browser)*

Gambar 2.2 Fitur-fitur *Web Apps*

2.5.3 Aplikasi *Mobile Hybrid*

Aplikasi *mobile hybrid* merupakan perpaduan antara aplikasi *mobile native* dan *mobile web* yang menggabungkan keunggulan dari keduanya. Fungsionalitas yang dimiliki aplikasi *mobile hybrid* lebih unggul dari pada pengembangan aplikasi *mobile web* dengan waktu pengembangan yang relatif cepat serta dengan biaya yang relatif murah dari pada aplikasi *mobile native. Mobile Hybrid* menggunakan pendekatan *cross-platform* yaitu dapat berkerja pada semua *platform* yang ada, dengan menggunakan teknologi *JavaScript* yang dapat dirancang sesuai keinginan sendiri ataupun menggunakan aplikasi *Open-source* yang menyediakan *Interface JavaScript* yang dapat digunakan sebagai jembatan yang menjaga konsistensi antar *Interface* pada setiap *Platform* (IBM Corporation, 2012).

Feature	***Native App***
Developmet language	*Native and web or web only*
Code portability and optimization	*High*
Access device-specific features	*Medium*
Leverage exixting knowledge	*High*
Advance graphics	*Medium*
Upgrade flexibility	*Medium (Usually by way of app stores)*
Installation experience	*High (From app store)*

Gambar 2.3 Fitur-fitur *Hybrid Apps*

2.6 *Android*

Android adalah sebuah *Operating System* (*OS*) untuk perangkat *mobile* berbasis Linux. OS Android mencakup sistem operasi, *middleware*, dan aplikasi. Android menyediakan *platform* terbuka untuk membuat aplikasi sendiri bagi para pengembang. Pada awalnya dikembangkan oleh *Android Inc*. *Android Inc.* adalah sebuah perusahaan pendatang baru yang membuat aplikasi untuk *smartphone* yang kemudian dibeli oleh *Google Inc* (Safaat H, 2012).

Android memiliki *platform* yang lengkap memungkinkan untuk para pengembang melakukan pengembangan *platform android* secara komprehensif, karena dalam *platform android* disini menyediakan *tools* yang lengkap dan fitur keamanannya. Seluruh *tools*, fitur dan lisensi yang ada pada *platform android* merupakan *Free Platform* dan *Open Source Platform* yang memberikan kebebasan kepada pengembang (Safaat H, 2012).

Android Development Tools (ADT) merupakan plugin yang digunakan dan dirancang untuk *Integrated Development Environtment (IDE)* yang memberikan kemudahan dalam mengembangkan aplikasi *Android*. *Tools* ini juga membantu membuat *Graphical User Interface (GUI)* pada aplikasi, menambahkan komponen *Android,* melakukan *running* aplikasi serta melakukan pembuatan *package android (.apk)* untuk mendistribusi aplikasi *Android* yang kita kembangkan (Safaat H, 2012).

2.6.1 *Android Software Development Kit (SDK)*

Salah satu *tools* yang digunakan adalah *Android Software Development Kit (SDK)* yang merupakan *tools* untuk *Application Programming Interface (API). Android SDK* ini diperlukan untuk dapat membantu mengembangkan aplikasi pada *platform Android* menggunakan bahasa pemrograman *Java* (Meier, 2012). Fitur-fitur yang ada pada *Android* yang penting adalah :

- *GSM, EDGE, 3G, 4G, and LTE networks,* untuk jaringan dalam menerima dan membuat telpon atau pesan singkat, serta menerima dan mengirim paket data dari jaringan.
- *Comprehensive APIs,* untuk layanan *APIs* komprehensif yang berbasis lokasi seperti *Global Position System (GPS)*.
- *Full support,* untuk dukungan aplikasi yang terintegrasi dengan peta kontrol sebagai antarmuka pengguna.
- *Wi-Fi hardware access and peer-to-peer, connections* untuk akses perangkat keras *Wi-Fi* dan koneksi *peer-to-peer.*
- *Full multimedia hardware control,* untuk kontrol perangkat keras multimedia seperti kamera dan mikrofon.
- *Media libraries,* untuk pemutaran media dengan berbagai format audio dan video atau gambar.
- *APIs for using sensor hardware,* untuk akses *APIs* dalam penggunaan seperti akselerometer, kompas, dan barometer.
- *Libraries for using Bluetooth and NFC hardware,* untuk penggunaan *Bluetooth* dan *NFC* dalam transfer data secara *peer-to-peer.*
- *IPC message*, untuk penerimaan pesan *IPC.*
- *Shared data stores and APIs*, untuk berbagi data dan *APIs* untuk kontak, jaringan sosial, kalender dan *multimedia*.
- *Background Services, applications, and processes,* untuk proses, servis, dan layanan.
- *Home-screen Widgets and Live Wallpaper,* untuk penampilan *widget* dan *wallpaper*.
- The ability to *Integrate application search,* kemampuan untuk integrasi hasil pencarian aplikasi pada pencarian sistem.
- *An integrated open-source HTML5 WebKit-based browser,* untuk integrasi browser *HTML5 WebKit Open Source.*
- *Mobile-optimized, hardware-accelerated graphics,* untuk optimalisasi kerja perangkat dan akselerasi grafik dengan perangkat keras.
- *Localization through a dynamic resource framework,* untuk pelokalan yang dinamis.
- *Application component reuse framework,* untuk penggunaan kembali komponen aplikasi.

2.6.2 *Android Sensor*

Pada *Android* terdapat *Sensor Manager* yang berfungsi untuk mengelola setiap penggunaan sensor-sensor yang disediakan oleh *Andorid.* Menggunakan perintah *getSystemService* untuk mengaktifkan penggunaan sensor yang ada pada *Sensor Manager Service.* Pada bagian berikut akan diberikan penjelasan tentang setiap jenis sensor yang ada pada *Android* yang dapat digunakan dalam aplikasi (Meier, 2012).

- *Temprature Sensor (Sensor.TYPE_TEMPERATURE),* merupakan sensor yang digunakan untuk mengukur temperatur atau suhu dalam derajat Celsius.
- *Accelerometer Sensor (Sensor.TYPE_ACCELEROMETER),* merupakan sensor yang digunakan untuk mengukur percepatan dalam m/s^2.
- *Gravity Sensor (Sensor.TYPE_GRAVITY),* merupakan sensor yang digunakan untuk mengukur percepatan gravitasi.
- *Linear Acceleration Sensor (Sensor.TYPE_LINEAR_ACCELERATION),* merupakan sensor yang digunakan untuk mengukur percepatan linier.
- *Gyroscope Sensor (Sensor.TYPE_GYROSCOPE),* merupakan sensor yang digunakan untuk mengukur rata-rata rotasi pada perangkat.
- *Rotation Vector Sensor (Sensor.TYPE_ROTATION_VECTOR),* sensor yang digunakan untuk mengukur rotasi vektor pada perangkat.
- *Magnetic Field Sensor (Sensor.TYPE_MAGNETIC_FIELD),* sensor yang digunakan untuk mengukur dan menemukan medan magnet disekitar perangkat.
- *Pressure Sensor (Sensor.TYPE_PRESSURE),* sensor yang digunakan untuk mengukur tekanan atmosfer dalam satuan *mbars*, dapat mengukur posisi ketinggian pada perangkat, serta dapat mengukur tekanan udara.
- *Relative Humidity Sensor (Sensor.TYPE_RELATIVE_HUMIDITY),* sensor yang digunakan untuk mengukur kelembaban udara.
- *Proximity Sensor (Sensor.TYPE_PROXIMITY),* sensor ini digunakan untuk mengukur jarak antara perangkat dengan objek tertentu yang telah ditentukan.
- *Light Sensor (Sensor.TYPE_LIGHT),* sensor ini digunakan untuk mengukur tingkat kecerahan cahaya dan untuk mengendalikan kecerahan layar secara dinamis.
- *Camera Sensor,* kamera pada perangkat *Android* termasuk sensor pada *Android* yang digunakan untuk mengambil gambar dan video untuk disimpan kedalam perangkat.
- *Global Positioning System (GPS), GPS* merupakan salah satu sensor yang digunakan untuk pengambilan informasi posisi dan letak perangkat.

2.7 System Development Life Cycle (SDLC)

System Development Life Cycle merupakan sebuah proses dimana memiliki serangkaian aktifitas yang saling berkaitan dalam membuat sebuah perangkat lunak, agar perangkat lunak lebih terstruktur dan mudah dipahami sehingga dapat menghasilkan perangkat lunak yang baik. Pada aktifitas-aktifitas yang terkait disini dapat dikembangkan dengan cara menambahkan, mengubah, atau mengadopsi dari sistem yang telah tersedia atau dengan mencampurkan beberapa komponen sistemnya (Sommerville, 2011).

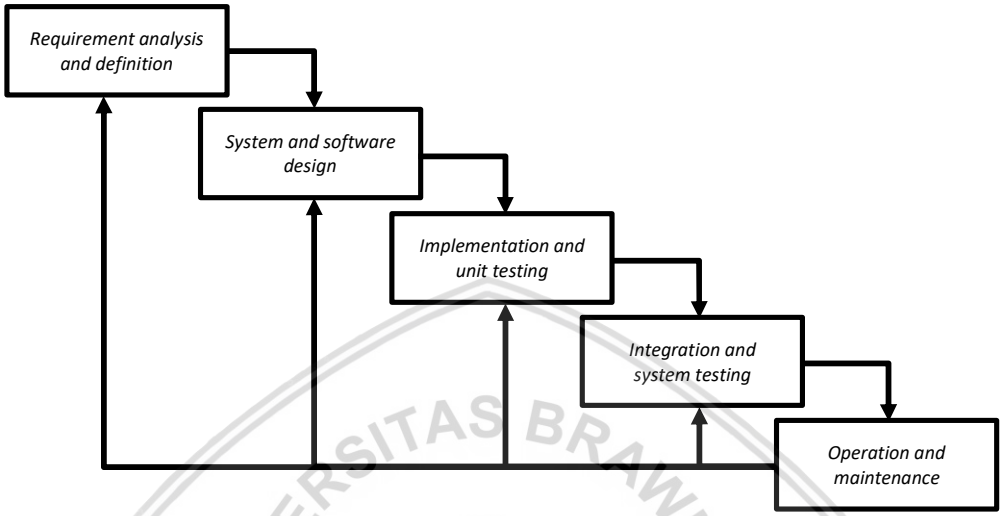
Terdapat beberapa proses atau aktifitas yang merupakan aktifitas dasar untuk melakukan perancangan perangkat lunak, diantaranya:

1. *Software Specification* : merupakan proses spesifikasi fungsional dari sistem yang harus didefinisikan secara jelas.
2. *Software Design and Implementation* : merupakan proses perancangan sistem dan implementasi perancangan sistem yang harus dilakukan.
3. *Software Validation* : merupakan proses validasi sistem agar perancangan sistem sesuai dengan spesifikasi awal pada perancangan sistem.
4. *Software Evolution* : merupakan proses pengembangan sistem agar tetap dapat memenuhi kebutuhan sistem.

Pengembangan perangkat lunak memiliki banyak *process model* yang dapat digunakan, salah satunya adalah *process model "waterfall". Waterfall model* merupakan sebuah proses yang pada dasarnya seperti air terjun yang mengalir, jadi proses perencanaan, penjadwalan, dan semua aktifitas proses harus didefinisikan terlebih dahhulu secara jelas dan terencana sebelum memulai proses perancangannya. Prinsip dasar pada pengembangan menggunakan *waterfall model* adalah sebagai berikut (Sommerville, 2011):

1. *Requirement analysis and definition* : merupakan suatu proses dimana kebutuhan sistem, tujuan sistem harus didefinisikan secara detail dan jelas yang didapatkan dengan melakukan konsultasi kepada pengguna sistem.
2. *System and software design* : merupakan suatu proses perancangan desain sistem dari kebutuhan-kebutuhan sistem yang telah didefinisikan sebelumnya, untuk membangun arsitektur sistem yang sesuai dengan sistem yang akan dirancang.
3. *Implementation and unit testing* : pada proses ini, desain sistem yang telah dirancang ebelumnya akan diimplementasikan dan diwujudkan sebagai rangkaian *program* atau *program unit.* Pada proses *Unit testing,* dilakukan verifikasi pada setiap unit bahwa setiap unit telah sesuai dengan kebutuhan diawal yang diinginkan
4. *Integration and system testing* : pada proses ini akan dilakukan integrasi pada setiap *program units* sebagai sebuah sistem yang utuh untuk memastikan bahwa kebutuhan sistem telah terpenuhi. Setelah sistem sesuai maka akan dilakukan testing kepada pengguna.

5. *Operation and maintenance* : pada proses ini merupakan proses perawatan dan perbaikan jika ada *program unit* yang bermasalah, agar sistem dapat tetap bekerja secara normal.



Gambar 2.4 *System Development Life Cycle (SDLC) Waterfall Model*

2.8 Java

Pada awal era 1990, pemrograman berorientasi objek menggunakan bahasa pemrograman *C++*, yang merupakan bahasa pemrograman yang dapat dikatakan sempurna pada eranya. Karena bahasa pemrograman *C++* memiliki efisiensi yang tinggi dan elemen-elemen yang lebih bagus dari bahasa pemrograman *C.* Namun itu sudah menjadi hal yang kuno, dan memaksa terjadinya evolusi pada pengembangan bahasa pemrograman (Schildt, 2007).

Bahasa pemrograman *Java* memiliki banyak kesamaan atau korelasi dengan bahasa pemrograman *C++* dimana merupakan keturunan bahasa pemrograman *C*. Banyak penurunan dari java yang berasal dari dua bahasa pemrograman tersebut, diantaranya *syntax* yang dimiliki *Java* merupakan penurunan dari bahasa *C.* Fitur-fitur yang berorientasi objek pada *Java* juga banyak dipengaruhi oleh bahasa *C++*. Kelahiran bahasa pemrograman *Java* berasalkan dari perubahan dan pengembangan serta perbaikan dari bahasa pemrograman yang terdahulu selama beberapa dekade. Setiap inovasi yang berada dalam bahasa pemrograman *Java* merupakan dorongan dari penyelesaian dan pemecahan masalah yang terjadi pada pemrograman dengan bahasa sebelumnya (Schildt, 2007).

Java dikembangkan *oleh James Gosling, Patrick Naughton, Chris Warth, Ed Frank dan Mike Sheridan* yang berada dalam 1 badan usaha yang sama yaitu *"Sun Microsystem".* Awalnya bahasa pemrograman *Java* dinamai *"Oak"* namun diganti *"Java"* pada tahun 1995. *Bill Joy, Arthur van Hoff, Jonathan Payne, Frank Yellin* dan *Tim Lindholm* juga ikut berjasa dalam pematangan prototipe asli dan kontribusi evolusi bahasa pemrograman. Motivasi utama dalam revolusi bahasa pemrograman disini adalah dibutuhkan nya *platform* yang berdiri sendiri dan dapat digunakan untuk membuat *software* untuk digunakan kedalam beberapa

perangkat elektronik seperti *microwave, remote controls,* dan perangkat lain yang sering digunakan (Schildt, 2007).

Kekuatan dasar penemuan bahasa pemrograman *Java* adalah *compatibility* dan *security. Compatibility* merupakan aspek utama pada aplikasi karena terdapat banyak jenis komputer atau perangkat dengan sistem operasi yang berbeda, maka dibutuhkan sebuah bahasa pemrograman yang dapat menghasilkan kode yang dapat berjalan pada semua jenis perangkat dan sistem operasi yang berbeda, dengan kode yang sama dan dapat menjamin keamanan dan kompabilitas. Pada *Security* menyediakan jaminan kemanan agar pencegahan dari serangan *virus* untuk mengakses *System Resource* dapat dihindari. *Java* memberikan proteksi dengan memberi batasan dalam eksekusi kode program (Schildt, 2007).

2.9 JavaScript

JavaScript merupakan bahasa pemrograman yang digunakan dalam berbagai *platform* misalkan pada *mobile*, *desktop*, *server*, *HTML* dan lainnya. Pada awalnya *JavaScript* diperkenalkan pada tahun 1995 sebagai program *browser Netscape Navigator* yang memiliki fitur interaktif terhadap penggunanya. Seiring perkembangan jaman *JavaScript* mengalami penyesuaian terhadap banyak *platform* pemrograman sehingga *JavaScript* dapat mendukung dalam pemrograman di berbagai *platform* (Haverbeke, 2018). *JavaScript* memiliki banyak bagian bahasa pemrograman, beberapa bagian tersebut akan dijelaskan pada Subbab 2.10.

2.9.1 *Nodejs*

Nodejs merupakan *platform software* yang ditulis dengan bahasa pemrograman *JavaScript* untuk komunikasi dan fungsi yang digunakan pada sisi *server. Nodejs* memungkinkan untuk menjalankan fungsi *JavaScript* diluar *browser* yang dirancang untuk menghubungkan node-node jaringan. Selain itu pada nodejs juga memiliki *library* yang banyak serta bermacam-macam pada modul *JavaScript* yang berguna untuk penyederhanaan pengembangan aplikasi (Haverbeke, 2018).

Fitur-fitur yang penting yang tersedia pada nodejs adalah (Tutorials Point Ltd., 2016):

a. *Asynchronous and Event Driven*

   Semua API dari *library nodejs* adalah *asynchronous* yang berarti pada dasarnya server yang berbasis *nodejs* tidah pernah menunggu API untuk mengembalikan data.

b. *Very Fast*

   Nodejs memiliki kecepatan dalam mengeksekusi kode untuk komunikasi dan pertukaran data pada jaringan.

c. *Single Threaded but Highly Scalable*

Nodejs menggunakan model *thread* tunggal dengan perulangan yang berarti membantu *server* untuk merespon permintaan dengan program yang sama dan dapat menyediakan layanan yang besar kepada *server.*

d. *No Buffering*

Nodejs tidak pernah melakukan *buffering data* karena *nodejs* hanya merupakan hasil keluaran dari potongan atau seleksi data.

e. *License*

Nodejs dirilis secara resmi oleh lisensi MIT.

2.9.2 *JavaScript Object Notation (JSON)*

JavaScript Object Notation yang disingkat *JSON* merupakan sebuah representasi standar yang terbuka untuk pertukaran data sebagai atribut yang memiliki suatu nilai. Pada awalnya berasal dari *JavaScript* untuk penggunaan aplikasi *web* sebagai alternatif dalam penggunan bahasa *XML. JSON* lebih terstruktur dan ringan dari bahasa *XML* dan menyediakan sarana ideal untuk mengenkapsulasi sirkulasi pengiriman dan penerimaan data yang terjadi antara *pengguna* dan *server* (Rischpater, 2015).

Pada *JSON* terdapat struktur penyusun yang merupakan kumpulan pasangan atribut yang memiliki nilai yang dapat dinyatakan sebagai *object, record, struct, dictionary, hash table, key list* atau *assiciate arrays.* Daftar atribut yang ber-ururtan yang dapat dinyatakan sebagai *array, vector, list,* atau *sequence.* Struktur-struktur data yang ada diatas merupakan struktur data universal. Semua bahasa pemrograman dapat mendukung struktur data ini, karena pada format data lebih mudah untuk dipahami dengan bahasa-baha pemrograman yang menggunakan struktur yang ada pada *JSON* (Ecma International, 2013).

Penulisan pada format *JSON* lebih sederhana dan lebih ringan karena ukuran data yang dihasilkan lebih kecil daripada bahasa *XML*. Pada *XML* terdapat penulisan *tag* yang berulang kali dan sama sehingga saat *script* tersimpan maka memakan cukup banyak kapasitas. Pada *JSON* lebih mengutamakan penulisan tanpa menggunakan *tag* sehingga lebih ringkas dan membuat kecepatan *loading transferi* lebih cepat menggunakan *JSON* dibanding *XML* (Rischpater, 2015).

Berikut adalah penulisan-penulisan yang ada pada *JSON* (Ecma International, 2013)*:*

a. Penulisan *Object* merupakan sepasang atribut yang memiliki nilai yang tidak terurutkan. Penulisan objek dimulai dengan kurung kurawal buka ({) dan diakhiri dengan kurung kurawal tutup (}). Setiap nama diikuti dengan titik dua (:) dan setiap pasangan atribut dipisahkan oleh koma (,).
b. *Array* merupakan kumpulan nilai yang ber-urutan. *Array* dimulai dengan kurung siku buka ([) dan diakhiri dengan kurung siku tutup (]). Setiap nilai dipisahkan oleh koma (,).

c. Nilai atau *value* dapat berupa sebuah *string* dalam tanda kutip ganda, angka, nilai *boolean (true, false, null)* atau sebuah *Object* atau sebuah *Array*. Struktur-struktur tersebut dapat disusun secara hierarki atau bertingkat.

d. *String* merupakan kumpulan dari nol atau lebih karakter *Unicode*, yang dibungkus dengan tanda kutip ganda. Di dalam *string* dapat digunakan *backslash escapes* "\" untuk membentuk karakter khusus. Sebuah karakter mewakili karakter tunggal pada *string*. *String* sangat mirip dengan string bahasa pemrograman *C* atau *Java*.

e. Angka adalah sangat mirip dengan angka dalam bahasa pemrograman *C* atau *Java*, kecuali format oktal dan heksadesimal tidak digunakan.

f. Spasi kosong (*whitespace*) dapat disisipkan di antara pasangan tanda-tanda tersebut, kecuali beberapa detil *encoding* yang secara lengkap dipaparkan oleh bahasa pemprograman yang bersangkutan.

2.10 Google Firebase

Google firebase merupakan penyedia *tools* dan infrastruktur untuk mempermudah pengembangan aplikasi yang dilakukan oleh para pengembang. Layanan ini meningkatkan layanan kepada para pengembang dengan memberi layanan umum seperti *database, backend, secure authentication, messaging* dan lainnya. Dengan adanya layanan ini para pengembang dapat menghemat kebutuhan dalam pengembangan aplikasi (Moroney, 2017).

Google firebase memiliki 3 pilar utama yaitu berkembang, tumbuh, dan menghasilkan sesuatu yang baik. Membangun aplikasi yang lebih baik dengan menggunakan *Cloud Firestore, Machine Learning Kit, Cloud Function, Authentication, Hosting, Cloud Storage,* dan *Realtime database.* Kemudian memperbaiki kualitas aplikasi dengan *Crashlytics, Test Lab,* dan *Performance Monitoring.* Selain itu, melakukan pertumbuhan bisnis dengan *Google Analytics, Predictions, Cloud Messaging, Alpha Beta Testing, Remote Config, Dynamic Links,* dan *App Indexing* (Google, 2018).

Dengan *Google Firebase API* semua layanan dan fitur dapat digunakan tidak hanya pada *android,* namun juga pada *iOS* atau *web application. Realtime database* merupakan fitur yang sangat bermanfaat karena proses pengambilan *data* terjadi sangat cepat dan efisien serta terorganisir. *Google firebase* sudah menyediakan *API* secara otomatis dan disimpan dalam format *JavaScript Object Notation (JSON)* dan dapat diakses dari semua *platform* (Singh, 2016).

Dalam penelitian ini menggunakan beberapa layanan yang disediakan oleh *Google Firebase* diantaranya :

2.10.1 *Authentication*

Auntentifikasi merupakan layanan yang disediakan untuk memberikan perizinan kepada pengguna yang telah diberikan izin untuk mengakses kedalam sistem. Layanan autentifikasi login dan registrasi yang disediakan *Google Firebase*

dapat melalui *Gmail, Github, Facebook, Twitter,* dan dapat memberikan layanan *login* sesuai dengan keinginan pengembang (Google, 2018).

2.10.2 *Cloud Storage*

Cloud Storage merupakan layanan yang disediakan untuk media penyimpanan basis data. Layanan ini dapat menyimpan dan menampilkan data gambar, *video*, *audio*, dan data yang lain secara langsung pada *SDK (Standard Development Kit)* pada perangkat pengguna. Pengguna yang mendapatkan hak akses melakukan proses pengunggahan dan pengunduhan secara *background process* pada sistem (Google, 2018).

2.10.3 *Realtime Database*

Realtime Database merupakan *database* yang berada pada *cloud hosting.* Data yang disimpan pada database menggunakan format *JavaScript Object Notation (JSON)* dan proses sinkronisasi dilakukan secara *realtime* kepada setiap pengguna sistem yang terdaftar. Jika pengembang membuat aplikasi *cross platform* dengan menggunakan *SDK Android, iOS,* atau *JavaScript,* semua pengguna menggunakan sebuah *Instance Realtime Database* yang telah terbagi. Kemudian pada setiap pengguna akan menerima *update* data terbaru secara otomatis. Pada *Realtime Database* terdapat fitur *database function* yang digunakan untuk menjalankan kode program pada sisi *backend* secara otomatis sebagai respon pada suatu fungsi yang dijalankan yang berhubungan dengan penggunaan *database* (Google, 2018).

2.10.4 *Crashlytics*

Crashlytics merupakan layanan yang melaporkan secara detail tentang eror atau kerusakan pada aplikasi. Setiap eror akan dilaporkan berdasarkan masalah yang sama, dan diurutkan berdasar tingkat urgensi yang akan mempengaruhi kerja sistem yang digunakan pengguna. Selain itu, pengembang juga mendapatkan log khusus untuk mempermudah dalam menyelasikan masalah yang menyebabkan aplikasi *crash* atau macet (Google, 2018).

2.11 Pengujian Perangkat Lunak

Pengujian perangkat lunak merupakan bagian paling penting saat aplikasi telah selesai dikerjakan. Sebelum digunakan oleh pengguna, aplikasi harus melewati tahap pengujian perangkat lunak, sehingga kekurangan dan kesalahan pada aplikasi dapat diminimalkan. Selain itu, tahap ini dilakukan untuk menguji hasil implementasi dan kesesuaian desain dan kebutuhan yang telah disebutkan pada awal proses perancangan. Pada proses pengujian termasuk bagian dari proses *software verification* dan *software validation* (Sommerville, 2011).

Software verification lebih mengarah kepada pengecekan pada kesempurnaan pada kebutuhan-kebutuhan fungsional maupun non-fungsional yang pada awal telah disebutkan. Pada *software validation* lebih kearah pengujian umum, yaitu kesesuaian dengan harapan pengguna. Pada fase testing disini, aplikasi diarahkan pada dua hal, yaitu *validation testing dan defect testing.*

Validation testing dilakukan dengan memberikan suatu *test case* dan jika sistem menghasilkan *output* yang benar, maka pengujian validasi berhasil dilakukan. *Defect testing* dilakukan dengan memberikan suatu *test case* yang didesain untuk menghasilkan *output* yang salah atau tidak diharapkan (Sommerville, 2011).

2.11.1 Pengujian Unit

Pengujian unit atau *unit testing* dilakukan sebagai pengujian unit yang merupakan bagian terkecil suatu aplikasi yang berupa komponen yang terdapat pada suatu *software*. Pengujian unit dilakukan untuk mengetahui pada setiap komponen *software* yang telah melalui proses implementasi telah sesuai dengan perancangan dan kebutuhan awal yang telah disebutkan. Pengujian unit akan menggunakan metode pengujian *whitebox* (Pressman, 2010).

Whitebox merupakan pengujian dengan memanfaatkan kasus uji dengan struktur yang berasal dari perancangan untuk menentukan kasus uji. Kemudian kasus uji tersebut akan diujikan kepada setiap jalur fungsi pada aplikasi sudah berjalan semestinya atau tidak. Selain itu juga menggunakan perhitungan *cyclomatic complexity* dalam menentukan kesesuaian kerja fungsi dan jalur atau *basis path* (Pressman, 2010).

2.11.2 Pengujian Validasi

Pengujian validasi atau *validation testing* dilakukan sebagai pengujian kebutuhan fungsional sistem yang berfokus pada *input output* pada sistem. Pengujian yang digunakan untuk menemukan *error* pada fungsi yang ada dan mengabaikan proses yang terjadi pada internal sistem. Pengujian validasi disini menggunakan metode *Blackbox testing* yang memungkinkan dalam menentukan kondisi pada suatu fungsi pada sistem. *Blackbox testing* merupakan pelengkap dari *Whitebox testing* (Pressman, 2010). *Blackbox testing* dilakukan dengan membuat kasus uji yang akan diujikan dan disesuaikan dengan keluaran valid dari sistem.

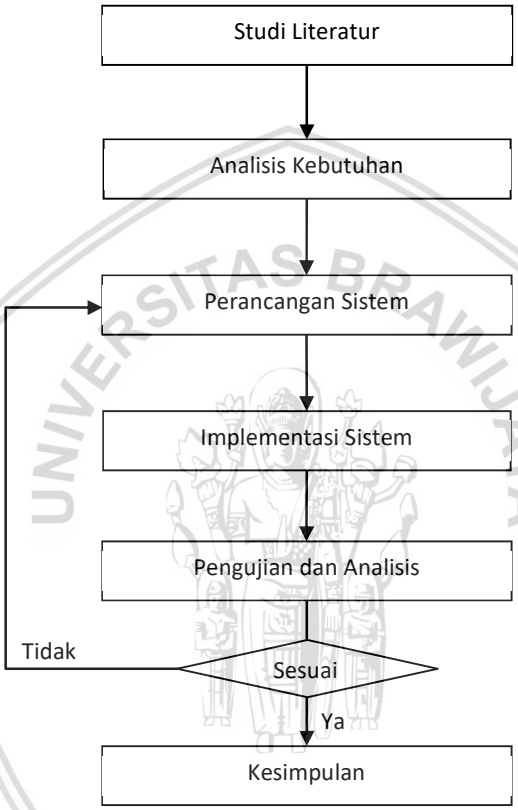
2.11.3 Pengujian *Compatibility*

Pengujian *Compatibility* merupakan salah suatu kegiatan pengujian pada aplikasi mobile yang ditujukan untuk memvalidasi fitur-fitur yang dibutuhkan oleh aplikasi di lingkunganyang berbeda. Lingkungan berbeda disini maksudnya adalah perangkat yang berbeda dari segi sistem operasi, ukuran layar dan sebagainya. Pengujian ini difokuskan pada tiga permasalahan utama yaitu *platform compatibility, device feature compatibility, native API compatibility* (Zhang, et al., March 2015).

Platform compatibility mengacu pada pengujian validasi aplikasi pada beberapa platform yang berbeda seperti *Android*, *iOS*, dan lainnya. *Device feature compatibility* mengacu pada pengujian validasi aplikasi pada beberapa fitur hardware yang berbeda seperti ukuran layar, RAM, CPU dan lainnya. *Native API compatibility* mengacu pada pengujian validasi aplikasi pada beberapa versi *API* yang berbeda (Zhang, et al., March 2015).

BAB 3 METODOLOGI

Bab ini menjelaskan metode atau tahapan yang digunakan dalam Pembangunan Aplikasi Informasi Kesehatan Masyarakat Kota Malang berbasis *Mobile Native Android*. Metode yang digunakan adalah metode *waterfall* dan tahapan yang digunakan meliputi studi literatur, analisis kebutuhan, perancangan sistem, implementasi sistem, pengujian dan analisis sistem, serta kesimpulan dari penelitian yang telah dilakukan, sebagaimana penjelasan tersebut dijelaskan pada Gambar 3.1.



Gambar 3.1 Diagram Alir Metodologi

3.1 Studi Literatur

Studi literatur merupakan cara untuk mempelajari dan memahami konsep yang akan diterapkan dalam perancangan sebuah sistem. Landasan-landasan teori yang menjadi bahan konsep dan dibutuhkan dalam menunjang kesuksesan penulisan skripsi ini menggunakan literatur yang bersumber pada skripsi, artikel ilmiah, penelitian sebelumnya, buku, dan informasi yang tersedia pada internet yang berbentuk proyek, penelitian, dan artikel. teori-teori yang dibutuhkan antara lain:

1. Pembangunan
2. Aplikasi Informasi Kesehatan masyarakat
3. Unified Modelling Language (UML)
4. Aplikasi Mobile

5. *Android*
6. *System Development Life Cycle (SDLC)*
7. *Java*
8. *Nodejs*
9. *JavaScript Object Notation (JSON)*
10. *Google Firebase*
11. Pengujian perangkat lunak

3.2 Analisis Kebutuhan

Analisis kebutuhan dalam perancangan aplikasi informasi kesehatan masyarakat, merupakan sebuah tahapan proses dalam menentukan aktifitas-aktifitas apa yang dapat dilakukan oleh aplikasi yang akan dirancang. Kebutuhan inilah yang kemudian diterapkan kepada pengguna aplikasi ini. Kebutuhan tersebut juga bisa kita dapatkan dari pengguna, maka dari itu pengumpulan data dari pengguna aplikasi sangat dibutuhkan, baik dari sisi *admin* dan pegawai biasa. Analisa kebutuhan dalam aplikasi ini akan dilakukan dengan metode wawancara yang dilakukan terhadap pengguna untuk melakukan analisis kebutuhan yang dibutuhkan oleh pengguna. Hasil dari wawancara yang dilakukan, akan direpresentasikan dengan *usecase diagram* yang merupakan kebutuhan pengguna yang telah disebutkan. Pada analisis kebutuhan disini terdapat kebutuhan fungsional sebagai kebutuhan yang penting dalam aplikasi, dan kebutuhan non-fungsional sebagai kebutuhan tambahan dalam aplikasi.

Kebutuhan fungsional merupakan kebutuhan penting yang harus tersedia agar aplikasi bisa memenuhi kebutuhan dan tujuan aplikasi, bagaimana perlakuan aplikasi saat ada data masukan, perlakuan aplikasi dalam situasi yang tertentu. Kebutuhan non-fungsional merupakan kebutuhan sampingan aplikasi yang diberikan kepada pengguna, yaitu kebutuhan yang tidak langsung terpusat pada kebutuhan yang diberikan dari aplikasi kepada pengguna. Contohnya seperti *Performance, availability,* dan *security.* Kebutuhan fungsional akan di modelkan dengan *usecase diagram* dan dijelaskan secara terperinci tentang alur kerjanya dengan *usecase scenario.*

3.3 Perancangan Aplikasi

Perancangan aplikasi ini akan dilaksanakan setelah semua kebutuhan yang telah diidentifikasi pada analisa kebutuhan aplikasi. Perancangan aplikasi dalam penelitian ini menggunakan metode *Object Oriented Design* dan menggunakan *Unified Modelling Language.* Hanya beberapa jenis *UML* yang diaplikasikan pada perancangan ini untuk mendukung proses perancangan aplikasi, macam-macam jenis *UML* tersebut adalah sebagai berikut

1. Perancangan Arsitektur aplikasi merupakan bagian awal dalam perancangan aplikasi dengan menunjukkan perancangan arsitektur aplikasi secara umum yang akan dikembangkan selanjutnya.

2. Perancangan *Interface* aplikasi merupakan perancangan antarmuka dalam perancangan aplikasi yang digunakan untuk merancang desain tampilan untuk interaksi antara pengguna dengan aplikasi.
3. Perancangan *Database* adalah perancangan struktur *database* untuk menentukan dan menggambarkan struktur penyimpanan data pada aplikasi.
4. Perancangan *Class Diagram* adalah diagram yang menunjukkan kelas apa saja yang ada pada aplikasi, dan relasi pada setiap kelas-kelas yang ada didalam aplikasi.
5. Perancangan *Activity Diagram* adalah diagram aktifitas yang menjelaskan alur atau aktifitas apa saja yang akan diaplikasikan pada perancangan aplikasi.
6. Perancangan *Sequence Diagram* adalah diagram sekuens yang menjelaskan tentang urutan proses yang terjadi untuk memenuhi tujuan dari *use case,* relasi atau hubungan antar kelas, operasi atau fungsi yang digunakan, urutan antar operasi atau fungsi serta informasi yang dibutuhkan sebagai kebutuhan aplikasi pada setiap operasi dan fungsi.

3.4 Implementasi

Implementasi atau penerapan aplikasi adalah langkah dimana untuk membangun aplikasi yang dilakukan setelah merampungkan tahap perancangan aplikasi. Tahap implementasi ini merubah hasil dari perancangan aplikasi menjadi bahasa pemrograman agar dapat bekerja sesuai dengan kepentingan pengguna. Implementasi pada aplikasi aplikasi ini akan dirancang berbasiskan *mobile* dengan aplikasi operasi *android.*

Pada tahap awal implementasi akan dilakukan proses desain *interface* atau antarmuka yang nyaman dan mudah untuk digunakan oleh pengguna dalam mendapatkan informasi. Implementasi *interface* akan dilakukan menggunakan bahasa *Extensible Markup Language* (*XML*) dengan menyesuaikan pada desain perancangan *interface* yang sebelumnya telah dirancang.

Pada tahap selanjutnya yaitu melakukan implementasi *database.* Implementasi pada aplikasi ini menggunakan *google firebase realtime database* sebagai simulasi penggunaan data Informasi Kesehatan Masyarakat. Setelah implementasi *interface* dan *database,* selanjutnya akan dilakukan implementasi kode program menggunakan bahasa *java* dengan memanfaatkan *software Android Studio.*

Bahasa pemrograman *Java* adalah bahasa pemrograman yang akan digunakan untuk merancang keseluruhan implementasi kode program aplikasi ini. Implementasi aplikasi ini secara keseluruhan akan dilakukan dengan menggunakan *Software Android Studio.*

3.5 Pengujian dan Analisis

Dalam tahap akhir perancangan yaitu tahap pengujian dan analisis merupakan tahap untuk memeriksa dan menemukan kesalahan dalam aplikasi yang diimplementasikan sebelumnya. Apakah aplikasi yang telah dibangun sesuai dengan kebutuhan, sesuai dengan tujuan dan layak untuk digunakan pengguna dalam memperoleh informasi tanpa mengalami kesulitan. Pengujian disini dilakukan dengan :

1. Pengujian Unit yaitu pengujian yang digunakan untuk menguji fungsi-fungsi beserta alogaritma-alogaritma yang digunakan dalam aplikasi ini yang telah diidentifikasikan pada proses perancangan. *Whitebox Testing* merupakan metode yang sesuai dalam pengujian semua algoritma dan fungsi yang ada.
2. Pengujian Validasi yaitu pengujian yang digunakan untuk menguji kebutuhan fungsional maupun non-fungsional, apakah aplikasi mampu memenuhi syarat-syarat aplikasi yang telah ditentukan sebelumnya. Pengujian Validasi ini dapat dilaksanakan dengan pemeriksaan aplikasi apakah sudah baik dan benar tidak ada *bug* atau *error* yang terjadi. Selain hal tersebut, dalam tahap ini juga dilakukan pengujian pada fitur-fitur yang ada dalam aplikasi. Dalam pengujian validasi dilakukan dengan metode *Blackbox Testing.*
3. Pengujian *Compatibility* yaitu pengujian yang digunakan untuk menguji kemampuan aplikasi untuk dipindahkan dari satu perangkat ke perangkat lainnya agar memiliki kemudahan dalam pengaksesan dan eksekusi aplikasi dalam beberapa perangkat berbeda.

3.6 Kesimpulan

Kesimpulan merupakan tahap yang dilakukan setelah melakukan pengujian serta analisis kepada penelitian. Kesimpulan yang diterima berdasar pada hasil pengujian dan analisis yang telah dilakukan sebelumnya. Dengan kesimpulan inilah akan diperoleh pokok bahasan dari hasil keseluruhan proses penelitian dan perancangan aplikasi. Lain dari pada hal tersebut juga kesimpulan memberikan informasi kepada pengembang untuk mengembangkan dalam pengembangan aplikasi.

BAB 4 ANALISIS KEBUTUHAN

Pada bab analisis kebutuhan disini akan dijelaskan tentang analisis kebutuhan pada aplikasi SIMKES yang dibagi menjadi empat bagian utama, yaitu gambaran umum tentang aplikasi, identifikasi setiap aktor yang akan terlibat pada aplikasi, analisis kebutuhan fungsional yang harus dipenuhi pada aplikasi serta kebutuhan non-fungsional sebagai kebutuhan pendukung pada pembangunan aplikasi ini. Pada fase analisis kebutuhan fungsional dibagi menjadi dua sub-bab. Kemudian terdapat dua *usecase* yaitu *usecase diagram* dan *usecase scenario.*

4.1 Gambaran Umum Aplikasi

Aplikasi SIMKES disini merupakan aplikasi yang berbasis *mobile native* yang dibangun untuk aplikasi informasi kesehatan masyarakat di Kota Malang. Aplikasi tersebut meliputi 3 fokus penelitian tentang informasi ibu hamil, penderita gizi buruk, dan penderita penyakit *tubercolosis*, yang dimana membutuhkan penanganan kesehatan secara cepat. Pada aplikasi ini memiliki fitur utama untuk mengelola data persebaran yang meliputi fungsi dalam mengedit, menambah, menghapus data gizi buruk. Menampilkan dan memetakan persebaran dari data ibu hamil, penderita gizi buruk, dan penderita *tubercolusis* dalam setiap kelurahan dan kecamatan Kota Malang.

Aplikasi ini dibangun dengan menggunakan *platform android* yang dikhususkan pada pembangunan aplikasi *mobile* dengan pendekatan secara *native,* kemudian pada pembangunan dengan pendekatan secara *native* menggunakan bahasa *java* sebagai bahasa pemrogramannya, dan menggunakan bahasa *Extensible Markup Language* (*XML*) untuk tampilan *layout* dan desain aplikasi. Sensor dalam android seperti sensor *Global Positioning System (GPS)* akan digunakan dalam pembangunan aplikasi ini. Dengan memanfaatkan sensor tersebut, maka kebutuhan aplikasi telah terpenuhi.

4.2 Identifikasi Aktor

Identifikasi aktor disini akan dijelaskan siapa saja yang akan berinteraksi dan dapat menggunakan fitur-fitur yang tersedia dalam aplikasi ini. Definisi aktor adalah orang yang terlibat dan dapat berinteraksi pada penggunaan aplikasi ini, berikut merupakan aktor yang dapat menggunakan fitur-fitur yang tersedia pada aplikasi yaitu *admin* dan pegawai. Penjelasan detail tentang a*dmin* dan pegawai akan dijelaskan pada Tabel 4.1.

Tabel 4.1 Tabel Identifikasi Aktor

No	Aktor	Deskripsi
1.	*Admin* (Dinas Kesehatan)	Merupakan pengguna pada aplikasi untuk melakukan tambah, hapus, edit, dan menampilkan persebaran penderita gizi buruk, menampilkan data ibu hamil, dan tubercolosis serta maps persebaran data.
2.	Pegawai (Puskesmas)	Merupakan pengguna pada aplikasi untuk melakukan tambah, hapus, serta edit data penderita gizi buruk, serta menampilkan data ibu hamil dan data *tubercolosis*.

4.3 Kebutuhan Fungsional

Pada sub bab kebutuhan fungsional akan dijelaskan tentang deskripsi dari kebutuhan-kebutuhan yang harus tersedia pada aplikasi informasi kesehatan masyarakat. Kebutuhan fungsional merupakan kebutuhan yang harus ada, sehingga dalam pembangunan aplikasi informasi kesehatan masyarakat dapat menjadi aplikasi yang benar dan sesuai dengan tujuan utama pembangunannya. Daftar kebutuhan fungsional akan dijelaskan secara rinci pada Tabel 4.2.

Tabel 4.2 Tabel Kebutuhan Fungsional

No.	Kode Fungsi	Nama Fungsi	Deskripsi
1	SIMKES-F01	*Login*	Aplikasi harus menyediakan fitur untuk login untuk dapat masuk kedalam aplikasi dan sebagai pemisah antara *admin* serta pegawai.
2	SIMKES-F02	*Logout*	Aplikasi harus menyediakan fitur untuk keluar setelah masuk kedalam aplikasi sebagai *admin* atau pegawai.
3	SIMKES-F03	Tampil *Data* Ibu Hamil	Aplikasi harus dapat menampilkan jenis data ibu hamil yang ada pada *database.*
4	SIMKES-F04	Tampil *Data* Balita	Aplikasi harus dapat menampilkan data balita yang telah disimpan pada *database.*
5	SIMKES-F05	*Add Data* Balita	Aplikasi harus dapat menambahkan data tunggal pada data balita.
6	SIMKES-F06	*Edit Data* Balita	Aplikasi harus dapat mengedit data tunggal pada data balita yang telah terdaftar pada database
7	SIMKES-F07	*Delete Data* Balita	Aplikasi harus dapat menghapus data tunggal pada data balita yang telah terdaftar pada database
8	SIMKES-F08	Tampil Data Gizi Buruk	Aplikasi harus dapat menampilkan jenis data gizi buruk yang telah disimpan pada *database.*
9	SIMKES-F09	*Add Data* Gizi Buruk	Aplikasi harus dapat menambahkan data tunggal pada jenis data gizi buruk.
10	SIMKES-F10	*Edit Data* Gizi Buruk	Aplikasi harus dapat menyunting data tunggal pada jenis data gizi buruk.
11	SIMKES-F11	*Delete Data* Gizi Buruk	Aplikasi harus dapat menghapus data tunggal pada jenis data gizi buruk.
12	SIMKES-F12	Tampil Data *Tubercolosis*	Aplikasi harus dapat menampilkan jenis data tubercolosis yang ada pada *database.*
13	SIMKES-F13	*Maps*	Aplikasi harus dapat menampilkan tingkat persebaran pada setiap jenis data yang akan ditampilkan dengan perbedaan skala warna pada peta kota malang

4.4 Kebutuhan Non-Fungsional

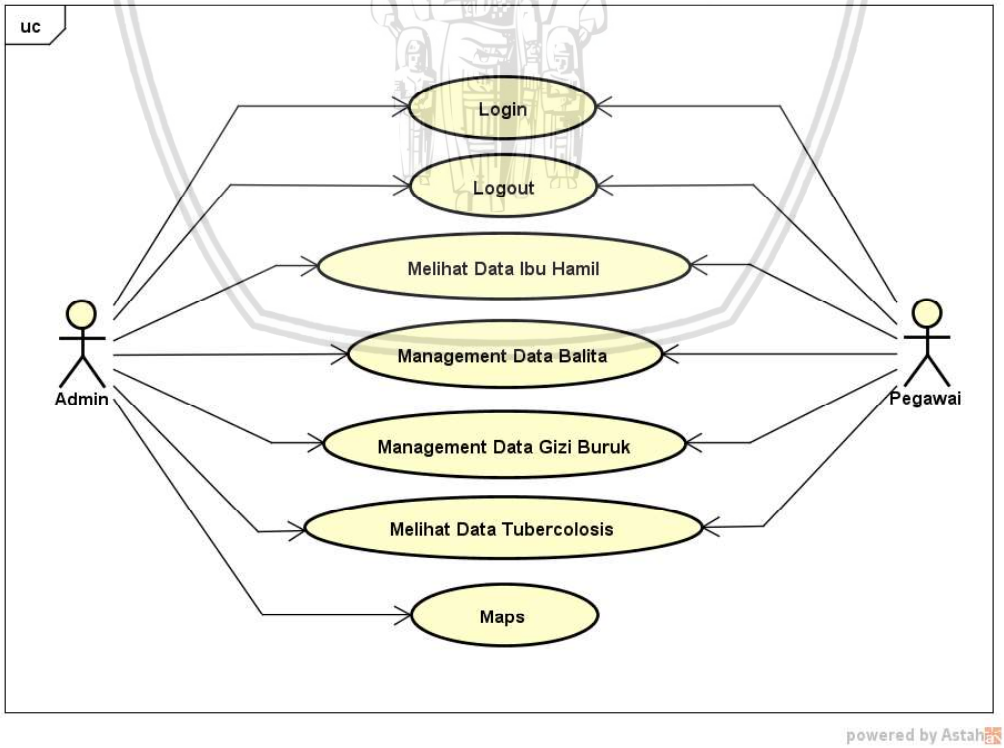
Pada kebutuhan non-fungsional akan dijelaskan tentang kebutuhan-kebutuhan apa saja yang akan mendukung kinerja kebutuhan fungsional yang merujuk pada Tabel 4.2. Kebutuhan non-fungsional merupakan kebutuhan yang dibutuhkan sebagai kebutuhan pendukung aplikasi dalam memenuhi kinerja aplikasi. Berikut merupakan kebutuhan non-fungsional yang terdapat pada Tabel 4.3:

Tabel 4.3 Tabel Kebutuhan Non-Fungsional

No.	Kode Fungsi	Nama Fungsi	Deskripsi
1	SIMKES-NF01	*Compatibility*	Aplikasi disarankan untuk memiliki kemudahan dalam pengaksesan dan eksekusi aplikasi dalam beberapa perangkat berbeda.

4.5 Usecase Diagram

Usecase Diagram merupakan diagram yang menggambarkan interaksi aktor dengan kebutuhan fungsional atau fungsi-fungsi utama yang disediakan pada aplikasi informasi kesehatan masyarakat. Identifikasi aktor dan kebutuhan fungsional yang juga telah disebutkan sebelumnya. Diagram *usecase* digambarkan dengan berawal dari kebutuhan fungsional sebagai kebutuhan yang akan dipetakan di dalam *usecase diagram* sebagai fungsi pada aplikasi informasi kesehatan masyarakat. Berikut merupakan pemetaan diagram usecase yang akan ditampilkan pada Gambar 4.1.



Gambar 4.1 Diagram *Usecase* SIMKES

4.6 Usecase Scenario

Pada sub-bab *usecase scenario* akan dijelaskan skenario dari setiap *usecase* yang telah disebutkan dan didefinisikan pada sub-bab *usecase diagram.* Dibawah ini akan dijelaskan *usecase scenario* pada Aplikasi Informasi Kesehatan Masyarakat.

4.6.1 *Usecase Scenario Login*

Usecase scenario pada proses *Login* dapat dilihat pada Tabel 4.4:

Tabel 4.4 *Usecase Scenario Login*

Kode use case	SIMKES-F01
Nama use case	*Login*
Tujuan	Fungsi agar pengguna dapat masuk kedalam aplikasi sebagai *admin* dan pegawai dapat menggunakan fitur-fitur pada aplikasi
Aktor	*Admin* dan pegawai
Pre-Condition	Pengguna membuka aplikasi dan telah berada pada interface awal aplikasi
Main Flow	1. *Admin* atau pegawai menekan tombol *login* 2. *Admin* atau pegawai mengisi *username* maksimal 50 karakter 3. *Admin* atau pegawai mengisi *password* maksimal 20 karakter 4. *Admin* atau pegawai menekan tombol login untuk konfirmasi data yang telah diisikan. 5. *Admin* atau pegawai menerima pemberitahuan bahwa telah masuk sebagai *admin* atau pegawai sesuai dengan *username* dan *password* yang telah diisikan
Alternative Flow	1. *Admin* atau pegawai tidak memasukkan *username*, dan aplikasi menampilkan pesan “Data inputan tidak sesuai” 2. *Admin* atau pegawai tidak memasukkan *password*, dan aplikasi menampilkan pesan “Data inputan tidak sesuai” 3. *Admin* atau pegawai tidak menekan tombol *login*, dan aplikasi menampilkan pesan “Data inputan tidak sesuai” 4. *Admin* atau pegawai memasukkan *username* yang salah, dan aplikasi menampilkan pesan “Data inputan tidak sesuai” 5. *Admin* atau pegawai memasukkan *password* yang salah, dan aplikasi menampilkan pesan “Data inputan tidak sesuai” 6. *Admin* atau pegawai memasukkan karakter *username* yang melebihi 50 karakter dan kurang dari 5 karakter, dan aplikasi menampilkan pesan “Data inputan tidak sesuai” 7. *Admin* atau pegawai memasukkan karakter *password* yang melebihi 20 karakter dan kurang dari 8 karakter, dan aplikasi menampilkan pesan “Data inputan tidak sesuai”
Post-Condition	Pengguna telah masuk kedalam aplikasi aplikasi

4.6.2 *Usecase Scenario Logout*

Usecase scenario pada proses *Logout* dapat dilihat pada Tabel 4.5:

Tabel 4.5 *Usecase Scenario Logout*

Kode use case	SIMKES-F02
Nama use case	*Logout*
Tujuan	Fungsi agar pengguna dapat keluar sebagai a*dmin* maupun pegawai dari aplikasi
Aktor	*Admin* dan pegawai
Pre-Condition	Pengguna sedang menggunakan aplikasi sebagai a*dmin* atau pegawai.
Main Flow	1. *Admin* atau pegawai menekan gambar profile 2. *Admin* atau pegawai menekan tombol *logout* 3. *Admin* atau pegawai mendapatkan pemberitahuan bahwa sudah keluar dari aplikasi
Alternative Flow	-
Post-Condition	*Admin* atau pegawai telah keluar dari aplikasi dan masuk pada halaman awal aplikasi

4.6.3 *Usecase Scenario* Tampil Data Ibu Hamil

Usecase Scenario pada proses Tampil Data Ibu Hamil dapat dilihat pada Tabel 4.6:

Tabel 4.6 *Usecase Scenario* Tampil *Data* Ibu Hamil

Kode use case	SIMKES-F03
Nama use case	Tampil Data Ibu Hamil
Tujuan	Fungsi agar *admin* atau pegawai dapat melihat informasi data dari ibu hamil
Aktor	*Admin* atau pegawai
Pre-Condition	Pengguna sedang menggunakan aplikasi sebagai a*dmin* atau pegawai
Main Flow	1. *Admin* atau pegawai menekan tombol tampil data ibu hamil 2. *Admin* atau pegawai mendapatkan informasi data ibu hamil
Alternative Flow	-
Post-Condition	*Admin* atau pegawai telah mendapatkan informasi data ibu hamil yang terdaftar pada database

4.6.4 *Usecase Scenario Management Data* Balita

Management Data Balita dibagi menjadi beberapa *Usecase Scenario* yaitu *Usecase Scenario* Tampil*, Add, Edit,* dan *Delete Data* Balita*.*

4.6.4.1 *Usecase Scenario* Tampil *Data* Balita

Usecase Scenario pada proses Tampil *Data* Balita dapat dilihat pada Tabel 4.7:

Tabel 4.7 *Usecase Scenario* Tampil *Data* Balita

Kode use case	SIMKES-F04
Nama use case	Tampil *Data* Balita
Tujuan	Fungsi agar *admin* atau pegawai dapat melihat informasi data balita

Aktor	*Admin* atau pegawai
Pre-Condition	Pengguna sedang menggunakan aplikasi sebagai a*dmin* atau pegawai
Main Flow	1. *Admin* atau pegawai menekan tombol tampil data balita 2. *Admin* atau pegawai mendapatkan informasi data balita
Alternative Flow	-
Post-Condition	*Admin* atau pegawai telah mendapatkan informasi balita yang terdaftar pada database

4.6.4.2 *Usecase Scenario Add Data* Balita

Usecase Scenario pada proses *Add Data* Balita dapat dilihat pada tabel 4.8:

Tabel 4.8 *Usecase Scenario Add Data* Balita

Kode use case	SIMKES-F05
Nama use case	*Add Data Balita*
Tujuan	Fungsi agar pegawai dapat mengunggah informasi data baru pada jenis data balita
Aktor	*Admin* dan Pegawai
Pre-Condition	Pengguna sedang menggunakan aplikasi sebagai pegawai dan berada pada halaman data gizi buruk
Main Flow	1. Pegawai menekan tombol tambah data balita 2. Pegawai mendapatkan form pengisian data balita 3. Pegawai mengisi nomor BPJS dengan tipe data angka maksimal 11 digit 4. Pegawai mengisi nama balita maksimal 25 karakter 5. Pegawai mengisi nama Orang Tua maksimal 25 karakter 6. Pegawai mengisi jenis kelamin balita dengan memilih pilihan yang telah disediakan aplikasi 7. Pegawai mengisi puskesmas dengan memilih pilihan puskesmas yang telah disediakan aplikasi 8. Pegawai menekan tombol simpan
Alternative Flow	1. Pegawai tidak mengisi nomor BPJS , dan aplikasi menampilkan pesan “Data inputan tidak sesuai” 2. Pegawai tidak mengisi nama balita atau lebih dari 25 karakter, dan aplikasi menampilkan pesan “Data inputan tidak sesuai” 3. Pegawai tidak mengisi nama Orang Tua atau lebih dari 25 karakter, dan aplikasi menampilkan pesan “Data inputan tidak sesuai” 4. Pegawai tidak mengisi jenis kelamin balita dengan memilih pilihan yang telah disediakan aplikasi, dan aplikasi menampilkan pesan “Data inputan tidak sesuai” 5. Pegawai tidak mengisi puskesmas dengan memilih pilihan puskesmas yang telah disediakan aplikasi, dan aplikasi menampilkan pesan “Data inputan tidak sesuai” 6. Pegawai menekan tombol *cancel* 7. Pegawai menekan tombol *close*
Post-Condition	Pengguna mendapatkan informasi data yang telah diunggah kedalam form tambah data balita

4.6.4.3 *Usecase Scenario Edit Data* Balita

Usecase Scenario pada proses *Edit Data* Balita dapat dilihat pada Tabel 4.9:

Tabel 4.9 *Usecase Scenario Edit Data* Balita

Kode use case	SIMKES-F06
Nama use case	*Edit Data* Balita
Tujuan	Fungsi agar pegawai dapat mengedit informasi data pada jenis data balita yang terdaftar pada aplikasi
Aktor	*Admin* dan Pegawai
Pre-Condition	Pengguna sedang menggunakan aplikasi sebagai pegawai dan berada pada halaman data gizi buruk
Main Flow	1. Pegawai menekan tombol edit data balita 2. Pegawai mendapat form pengisian data balita 3. Pegawai mengisi nomor BPJS dengan tipe data angka maksimal 11 digit 4. Pegawai mengisi nama balita maksimal 25 karakter 5. Pegawai mengisi nama Orang Tua maksimal 25 karakter 6. Pegawai mengisi jenis kelamin balita dengan memilih pilihan yang telah disediakan aplikasi 7. Pegawai mengisi puskesmas dengan memilih pilihan puskesmas yang telah disediakan aplikasi 8. Pegawai menekan tombol simpan
Alternative Flow	1. Pegawai tidak mengisi nomor BPJS, dan aplikasi menampilkan pesan “Data inputan tidak sesuai” 2. Pegawai tidak mengisi nama balita atau lebih dari 25 karakter, dan aplikasi menampilkan pesan “Data inputan tidak sesuai” 3. Pegawai tidak mengisi nama Orang Tua atau lebih dari 25 karakter, dan aplikasi menampilkan pesan “Data inputan tidak sesuai” 4. Pegawai tidak mengisi jenis kelamin balita dengan memilih pilihan yang telah disediakan aplikasi, dan aplikasi menampilkan pesan “Data inputan tidak sesuai” 5. Pegawai tidak mengisi puskesmas dengan memilih pilihan puskesmas yang telah disediakan aplikasi, dan aplikasi menampilkan pesan “Data inputan tidak sesuai” 6. Pegawai menekan tombol *cancel* 7. Pegawai menekan tombol *close*
Post-Condition	Pengguna mendapatkan informasi data yang telah diupdate kedalam form tambah data balita

4.6.4.4 *Usecase Scenario Delete Data* Balita

Usecase Scenario pada proses *Delete Data* Balita dapat dilihat pada Tabel 4.10:

Tabel 4.10 *Usecase Scenario Delete Data Balita*

Kode use case	SIMKES-F07
Nama use case	*Delete Data* Balita
Tujuan	Fungsi agar pegawai dapat menghapus informasi data tunggal pada jenis data balita yang telah terdaftar pada aplikasi

Aktor	*Admin* dan Pegawai
Pre-Condition	Pengguna sedang menggunakan aplikasi sebagai pegawai dan berada pada halaman data balita
Main Flow	1. Pegawai menekan tombol *delete data* balita 2. Pegawai mendapatkan konfirmasi untuk menghapus data balita yang telah terdaftar pada aplikasi 3. Pegawai mendapatkan informasi bahwa proses *delete data* balita telah berhasil
Alternative Flow	1. Pegawai menekan tombol *cancel*
Post-Condition	Pengguna telah mendapatkan informasi yang terbaru dari daftar data balita yang terdaftar pada aplikasi

4.6.5 *Usecase Scenario Management Data* Gizi Buruk

Management Data Gizi Buruk dibagi menjadi beberapa U*secase Scenario* yaitu *Usecase Scenario* Tampil*, Add, Edit,* dan *Delete Data* Gizi Buruk*.*

4.6.5.1 *Usecase Scenario* Tampil *Data* Gizi Buruk

Usecase Scenario pada proses Tampil *Data* Gizi Buruk dapat dilihat pada Tabel 4.11:

Tabel 4.11 *Usecase Scenario* Tampil *Data* Gizi Buruk

Kode use case	SIMKES-F08
Nama use case	Tampil *Data* Gizi Buruk
Tujuan	Fungsi agar *admin* atau pegawai dapat melihat informasi data dari penderita gizi buruk
Aktor	*Admin* atau pegawai
Pre-Condition	Pengguna sedang menggunakan aplikasi sebagai a*dmin* atau pegawai
Main Flow	1. *Admin* atau pegawai menekan tombol tampil data gizi buruk 2. *Admin* atau pegawai mendapatkan informasi data gizi buruk
Alternative Flow	-
Post-Condition	*Admin* atau pegawai telah mendapatkan informasi data gizi buruk yang terdaftar pada database

4.6.5.2 *Usecase Scenario Add Data* Gizi Buruk

Usecase Scenario pada proses A*dd Data* Gizi Buruk dapat dilihat pada Tabel 4.12:

Tabel 4.12 *Usecase Scenario Add Data* Gizi Buruk

Kode use case	SIMKES-F09
Nama use case	*Add Data* Gizi Buruk
Tujuan	Fungsi agar pegawai dapat mengunggah informasi data baru pada jenis data gizi buruk
Aktor	*Admin* dan Pegawai
Pre-Condition	Pengguna sedang menggunakan aplikasi sebagai pegawai dan berada pada halaman data gizi buruk
Main Flow	1. Pegawai menekan tombol *add data* gizi buruk pada halaman informasi data gizi buruk

	2. Pegawai mendapatkan form pengisian data gizi buruk 3. Pegawai mengisi kolom tanggal dengan tipe data tanggal 4. Pegawai mengisi kolom puskesmas dengan memilih daftar puskesmas yang terdaftar pada aplikasi 5. Pegawai mengisi kolom kelurahan dengan memilih daftar kelurahan yang terdaftar pada aplikasi 6. Pegawai mengisi kolom nomor kelurahan dengan mengisi nomor kelurahan dengan tipedata angka maksimal 11 digit 7. Pegawai mengisi kolom balita dengan mengetik dan memilih nama balita yang sudah terdaftar pada aplikasi. 8. Pegawai mengisi alamat maksimal 45 karakter 9. Pegawai mengisi kolom RT dengan tipe data angka maksimal 11 karakter 10. Pegawai mengisi kolom RW dengan tipe data angka maksimal 11 karakter 11. Pegawai mengisi kolom umur dengan tipe data angka maksimal 45 digit dalam satuan bulan 12. Pegawai mengisi kolom berat badan dengan tipe data angka maksimal 11 digit dalam satuan kilogram 13. Pegawai mengisi kolom tinggi badan dengan tipe data angka maksimal 11 digit dalam satuan centimeter 14. Pegawai mengisi kolom bulan masuk dengan tipe data angka maksimal 11 digit 15. Pegawai mengisi kolom bulan keluar dengan tipe data angka maksimal 11 digit 16. Pegawai mencentang opsi GAKIN jika keluarga penderita gizi buruk adalah keluarga miskin 17. *Admin* atau pegawai menekan tombol simpan
Alternative Flow	1. Pegawai tidak mengisi kolom tanggal, dan aplikasi menampilkan pesan “Data inputan tidak sesuai” 2. Pegawai tidak mengisi kolom puskesmas, dan aplikasi menampilkan pesan “Data inputan tidak sesuai” 3. Pegawai tidak mengisi kolom kelurahan , dan aplikasi menampilkan pesan “Data inputan tidak sesuai” 4. Pegawai tidak mengisi kolom nomor kelurahan atau melebihi 11 digit, dan aplikasi menampilkan pesan “Data inputan tidak sesuai” 5. Pegawai tidak mengisi dan memilih kolom balita, dan aplikasi menampilkan pesan “Data inputan tidak sesuai” 6. Pegawai tidak mengisi alamat atau melebihi 45 karakter, dan aplikasi menampilkan pesan “Data inputan tidak sesuai” 7. Pegawai tidak mengisi kolom RT atau melebihi 11 karakter, dan aplikasi menampilkan pesan “Data inputan tidak sesuai” 8. Pegawai tidak mengisi kolom RW atau melebihi 11 karakter, dan aplikasi menampilkan pesan “Data inputan tidak sesuai” 9. Pegawai tidak mengisi kolom umur atau melebihi 45 digit, dan aplikasi menampilkan pesan “Data inputan tidak sesuai” 10. Pegawai tidak mengisi kolom berat badan atau melebihi 11 digit , dan aplikasi menampilkan pesan “Data inputan tidak sesuai” 11. Pegawai tidak mengisi kolom tinggi badan atau melebihi 11 digit , dan aplikasi menampilkan pesan “Data inputan tidak sesuai”

	12. Pegawai tidak mengisi kolom bulan masuk atau melebihi 11 digit, dan aplikasi menampilkan pesan “Data inputan tidak sesuai” 13. Pegawai tidak mengisi kolom bulan keluar atau melebihi 11 digit, dan aplikasi menampilkan pesan “Data inputan tidak sesuai” 14. Pegawai menekan tombol close 15. Pegawai menekan tombol batal
Post-Condition	Pengguna mendapatkan informasi data dan foto yang telah diunggah kedalam form tambah data gizi buruk

4.6.5.3 *Usecase Scenario Edit Data* Gizi Buruk

Usecase Scenario pada proses *Edit Data* Gizi Buruk dapat dilihat pada Tabel 4.13:

Tabel 4.13 *Usecase Scenario Edit Data* Gizi Buruk

Kode use case	SIMKES-F10
Nama use case	*Edit Data* Gizi Buruk
Tujuan	Fungsi agar pegawai dapat mengedit informasi data tunggal pada jenis data gizi buruk yang telah terdaftar pada aplikasi
Aktor	*Admin* dan Pegawai
Pre-Condition	Pengguna sedang menggunakan aplikasi sebagai pegawai dan berada pada halaman data gizi buruk
Main Flow	1. Pegawai menekan tombol *edit data* gizi buruk pada halaman informasi data gizi buruk 2. Pegawai mendapatkan form pengisian data gizi buruk 3. Pegawai mengisi kolom tanggal dengan tipe data tanggal 4. Pegawai mengisi kolom puskesmas dengan memilih daftar puskesmas yang terdaftar pada aplikasi 5. Pegawai mengisi kolom kelurahan dengan memilih daftar kelurahan yang terdaftar pada aplikasi 6. Pegawai mengisi kolom nomor kelurahan dengan mengisi nomor kelurahan dengan tipedata angka maksimal 11 digit 7. Pegawai mengisi kolom balita dengan mengetik dan memilih nama balita yang sudah terdaftar pada aplikasi. 8. Pegawai mengisi alamat maksimal 45 karakter 9. Pegawai mengisi kolom RT dengan tipe data angka maksimal 11 karakter 10. Pegawai mengisi kolom RW dengan tipe data angka maksimal 11 karakter 11. Pegawai mengisi kolom umur dengan tipe data angka maksimal 45 digit dalam satuan bulan 12. Pegawai mengisi kolom berat badan dengan tipe data angka maksimal 11 digit dalam satuan kilogram 13. Pegawai mengisi kolom tinggi badan dengan tipe data angka maksimal 11 digit dalam satuan centimeter 14. Pegawai mengisi kolom bulan masuk dengan tipe data angka maksimal 11 digit 15. Pegawai mengisi kolom bulan keluar dengan tipe data angka maksimal 11 digit 16. Pegawai mencentang opsi GAKIN jika keluarga penderita gizi buruk adalah keluarga miskin

	17. *Admin* atau pegawai menekan tombol simpan
Alternative Flow	1. Pegawai tidak mengisi kolom tanggal, dan aplikasi menampilkan pesan “Data inputan tidak sesuai” 2. Pegawai tidak mengisi kolom puskesmas , dan aplikasi menampilkan pesan “Data inputan tidak sesuai” 3. Pegawai tidak mengisi kolom kelurahan , dan aplikasi menampilkan pesan “Data inputan tidak sesuai” 4. Pegawai tidak mengisi kolom nomor kelurahan atau melebihi 11 digit, dan aplikasi menampilkan pesan “Data inputan tidak sesuai” 5. Pegawai tidak mengisi dan memilih kolom balita 6. Pegawai tidak mengisi alamat atau melebihi 45 karakter, dan aplikasi menampilkan pesan “Data inputan tidak sesuai” 7. Pegawai tidak mengisi kolom RT atau melebihi 45 karakter, dan aplikasi menampilkan pesan “Data inputan tidak sesuai” 8. Pegawai tidak mengisi kolom RW atau melebihi 45 karakter, dan aplikasi menampilkan pesan “Data inputan tidak sesuai” 9. Pegawai tidak mengisi kolom umur atau melebihi 45 digit, dan aplikasi menampilkan pesan “Data inputan tidak sesuai” 10. Pegawai tidak mengisi kolom berat badan atau melebihi 11 digit, dan aplikasi menampilkan pesan “Data inputan tidak sesuai” 11. Pegawai tidak mengisi kolom tinggi badan atau melebihi 11 digit, dan aplikasi menampilkan pesan “Data inputan tidak sesuai” 12. Pegawai tidak mengisi kolom bulan masuk atau melebihi 11 digit, dan aplikasi menampilkan pesan “Data inputan tidak sesuai” 13. Pegawai tidak mengisi kolom bulan keluar atau melebihi 11 digit, dan aplikasi menampilkan pesan “Data inputan tidak sesuai” 14. Pegawai menekan tombol close 15. Pegawai menekan tombol batal
Post-Condition	Pengguna mendapatkan informasi data dan foto yang telah diunggah dan diedit pada form tambah data gizi buruk

4.6.5.4 *Usecase Scenario Delete Data* Gizi buruk

Usecase Scenario pada proses *Delete Data* Gizi Buruk dapat dilihat pada Tabel 4.14:

Tabel 4.14 *Usecase Scenario Delete Data* Gizi Buruk

Kode use case	SIMKES-F11
Nama use case	*Delete Data* Gizi Buruk
Tujuan	Fungsi agar pegawai dapat menghapus informasi data tunggal pada jenis data gizi buruk yang telah terdaftar pada aplikasi
Aktor	*Admin* dan Pegawai
Pre-Condition	Pengguna sedang menggunakan aplikasi sebagai pegawai dan berada pada halaman data gizi buruk
Main Flow	1. Pegawai menekan tombol *delete data gizi buruk* 2. Pegawai mendapatkan konfirmasi untuk menghapus data gizi buruk yang telah terdaftar pada aplikasi

	3. Pegawai mendapatkan informasi bahwa proses *delete data gizi buruk* telah berhasil
Alternative Flow	1. Pegawai menekan tombol *cancel*
Post-Condition	Pengguna telah mendapatkan informasi yang terbaru dari daftar data gizi buruk yang terdaftar pada aplikasi

4.6.6 *Usecase Scenario* Tampil Data *Tubercolosis*

Usecase Scenario pada proses Tampil *Data Tubercolosis* dapat dilihat pada Tabel 4.15:

Tabel 4.15 *Usecase Scenario* Tampil *Data Tubercolosis*

Kode use case	SIMKES-F12
Nama use case	Tampil *Data Tubercolosis*
Tujuan	Fungsi agar *admin* atau pegawai dapat melihat informasi data dari penderita *tubercolosis*
Aktor	*Admin* atau pegawai
Pre-Condition	Pengguna sedang menggunakan aplikasi sebagai a*dmin* atau pegawai
Main Flow	1. *Admin* atau pegawai menekan tombol tampil data *tubercolosis* 2. *Admin* atau pegawai mendapatkan informasi data *tubercolosis*
Alternative Flow	-
Post-Condition	*Admin* atau pegawai telah mendapatkan informasi *tubercolosis* yang terdaftar pada database

4.6.7 *Usecase Scenario* Maps

Usecase Scenario pada proses *Maps* dapat dilihat pada Tabel 4.16:

Tabel 4.16 *Usecase Scenario* Tampil *Maps*

Kode use case	SIMKES-F13
Nama use case	*Maps*
Tujuan	Fungsi agar *admin* dapat menampilkan *Maps* yang menunjukkan persebaran setiap jenis data pada setiap daerah Kota Malang dengan parameter tertentu yang dipilih oleh pengguna
Aktor	*Admin*
Pre-Condition	Pengguna sedang menggunakan aplikasi sebagai *admin* atau pegawai dan berada pada halaman *maps*
Main Flow	1. *Admin* menekan tombol *Maps* 2. *Admin* beralih pada halaman *Maps* 3. *Admin* mendapatkan form pengisian *Maps* dan diisi dengan pilihan jenis data 4. *Admin* mengisi parameter yang ingin ditemukan berdasarkan pilihan yang disediakan oleh aplikasi 5. *Admin* memilih tampilkan *Maps*
Alternative Flow	1. *Admin* tidak mengisi parameter sesuai ketentuan aplikasi 2. *Admin* menekan tombol kembali
Post-Condition	Pengguna telah mendapatkan *Maps* persebaran data pada setiap daerah Kota Malang yang ingin dilihat oleh pengguna

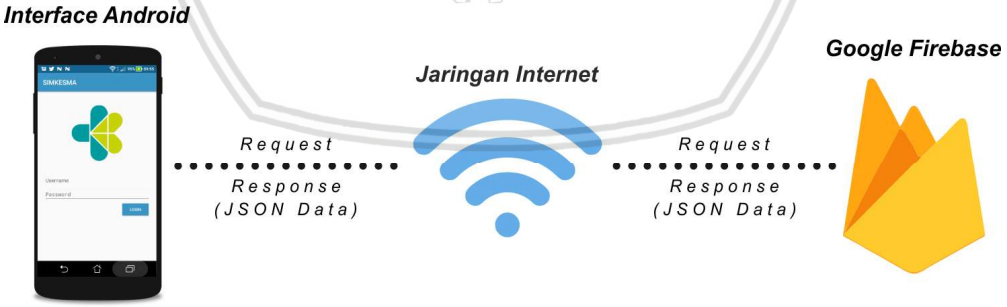
BAB 5 PERANCANGAN

Pada bab perancangan aplikasi informasi kesehatan masyarakat dilakukan berdasarkan analisis kebutuhan yang telah dilakukan. Perancangan aplikasi dimulai dengan perancangan arsitektur, perancangan *Interface*, perancangan *Database*, *Class Diagram, Activity Diagram,* dan *Sequence Diagram,* serta *Class Diagram*.

5.1 Perancangan Arsitektur

Pada perancangan arsitektur aplikasi merupakan bagian awal dalam melakukan perancangan pada aplikasi informasi kesehatan masyarakat yang akan digambarkan secara umum dan akan dikembangkan kedalam tahap perancangan selanjutnya.

Pada penelitian ini, aplikasi akan dirancang menggunakan arsitektur - *frontend* serta *backend* sehingga perancangan *interface* dan perancangan *database* serta pengolahan data lebih mudah dikembangkan. Perancangan arsitektural disini terdiri dari perancangan arsitektur pada perangkat *mobile* dan perancangan arsitektur *database.* Tentunya untuk perancangan yang terpisah membutuhkan penghubung untuk saling mengolah dan memproses data yang akan disimpan pada *database google firebase*. Komunikasi antara pengguna dengan *database* akan terlaksana jika terdapat *API* sebagai penghubung. Penggunaan *API* dapat menangani pertukaran dan komunikasi berupa *request* data yang terjadi pada pengguna dan *server*. Melalui *API* yang dimiliki serta dengan *format data* berupa *JSON* yang merupakan *format data* yang berisi data yang diterima pengguna dalam komunikasi pengguna dan *server* seperti yang ditunjukkan pada Gambar 5.1.



Gambar 5.1 Perancangan Arsitektur Aplikasi

Aplikasi ini pada sisi pengguna melakukan pertukaran data pada *Google firebase* dengan menggunakan *request API* yang telah disediakan oleh *Google firebase* dan direspon dalam bentuk data format *JSON.* Data pada *Google firebase* diproses dalam file yang kemudian akan di-*encode* menjadi format *JSON* dan dikirimkan pada pengguna dan ditampilkan pada *interface* aplikasi. Data yang dikirimkan oleh *Google firebase* yang berupa format *JSON* akan di-*decode* oleh aplikasi dan akan diolah lebih lanjut.

5.2 Perancangan *Interface*

Perancangan *interface* dirancang sebagai penghubung antara pengguna dan aplikasi yang berfungsi untuk menerima input dari pengguna untuk diproses selanjutnya oleh aplikasi. Berikut ini merupakan penjelasan perancangan *Interface* pada aplikasi pelayanan kesehatan masyarakat.

5.2.1 Perancangan Halaman *Login*

Gambar 5.2 Halaman *Login*

Pada Gambar 5.2 menggambarkan perancangan *interface* halaman *Login*, penjelasan lebih rinci akan dijelaskan pada Tabel 5.1:

Tabel 5.1 Penjelasan Halaman *Login*

No.	Nama Objek	Tipe	Keterangan
1.	Username	Field Box	Kotak inputan pengguna yang berupa username
2.	*Password*	*Field box*	Kotak inputan pengguna yang berupa password
3.	*Login*	*Button*	Tombol untuk mengkonfirmasi inputan yang terlah dimasukkan pengguna agar dapat diproses oleh aplikasi

5.2.2 Perancangan Halaman *Home*

Gambar 5.3 Halaman *Home*

Pada Gambar 5.3 menggambarkan perancangan *interface* halaman *Home*, penjelasan lebih rinci akan dijelaskan pada Tabel 5.2:

Tabel 5.2 Penjelasan halaman *Home*

No	Nama Objek	Tipe	Keterangan
1.	*Navigation Menu*	*Button*	Tombol untuk navigasi menu pada aplikasi
2.	Data Ibu Hamil	*Button*	Tombol untuk membuka detail data tentang ibu hamil
3.	Data Gizi Buruk	*Button*	Tombol untuk membuka detail data tentang gizi buruk
4.	Data Balita	*Button*	Tombol untuk membuka detail data tentang balita
5.	Data *Tubercolosis*	*Button*	Tombol untuk membuka detail data tentang *tubercolosis*

5.2.3 Perancangan *Navigation Menu*

Gambar 5.4 *Navigation Menu*

Pada Gambar 5.4 menggambarkan perancangan *interface Navigation Menu*, penjelasan lebih rinci akan dijelaskan pada Tabel 5.3:

Tabel 5.3 Penjelasan *Navigation Menu*

No	Nama Objek	Tipe	Keterangan
1.	*Navigation Menu*	*Button*	Tombol untuk navigasi menu pada aplikasi
2.	Logo	*Picture*	Gambar logo dari aplikasi
3.	*Username*	*Text View*	Teks yang berupa *username* yang sedang digunakan
4.	Daftar Menu	*Button*	Tombol untuk membuka menu-menu yang terdapat pada aplikasi
5.	*Sign Out*	*Button*	Tombol untuk keluar dari aplikasi

5.2.4 Perancangan Halaman *Management Data*

5.2.4.1 Perancangan Halaman Tampil Data

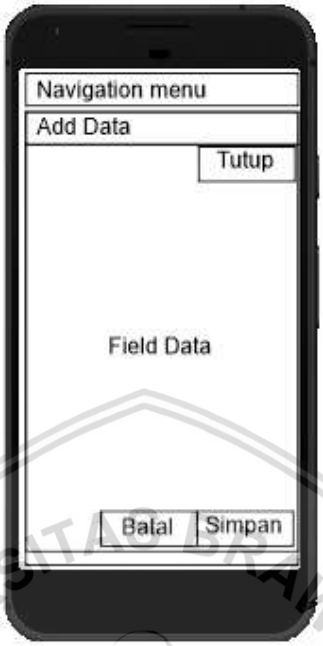
Gambar 5.5 Halaman Tampil *Data*

Pada Gambar 5.5 menggambarkan perancangan *interface* halaman Tampil Data, penjelasan lebih rinci akan dijelaskan pada Tabel 5.4:

Tabel 5.4 Penjelasan halaman Tampil Data

No	Nama Objek	Tipe	Keterangan
1.	*Navigation Menu*	*Button*	Tombol untuk navigasi menu pada aplikasi
2.	*Management Data*	*Text View*	Teks yang menerangkan halaman yang sedang aktif digunakan pengguna
3.	*Show Entries*	*Drop Down*	Tombol pilihan jumlah data yang akan ditampilkan kepada pengguna yang telah ditentukan oleh aplikasi
4.	Tambah Data	*Button*	Tombol untuk menjalankan fungsi menambahkan pengguna yang dapat mengakses kedalam aplikasi
5.	*Search*	*Field Text Box*	Tombol untuk inputan pengguna untuk mencari data yang ingin dicari
6.	*List Data*	*List View*	Tabel yang berisi data pengguna yang bisa mengakses aplikasi
7.	Nomor Halaman	Button	Tombol yang menampilkan halaman dan digunakan untuk berpindah halaman pada list data pengguna

5.2.4.2 Perancangan Halaman *Add Data*



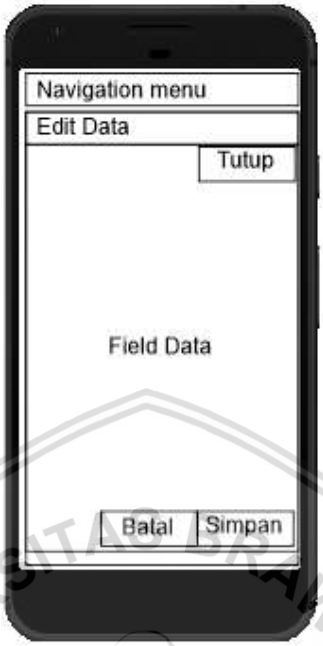
Gambar 5.6 Halaman *Add Data*

Pada Gambar 5.6 menggambarkan perancangan *interface* halaman *Add Data*, penjelasan lebih rinci akan dijelaskan pada Tabel 5.5:

Tabel 5.5 Penjelasan halaman *Add Data*

No	Nama Objek	Tipe	Keterangan
1.	*Navigation Menu*	*Button*	Tombol untuk navigasi menu pada aplikasi
2.	*Add data*	*Text View*	Teks yang menerangkan halaman yang sedang aktif digunakan pengguna
3.	*Field Data*	*Data Form*	Form untuk mengisi data pengguna baru yang dapat mengakses aplikasi
4.	Tutup	*Button*	Tombol untuk menutup form data pengisian pengguna baru
5.	Batal	*Button*	Tombol untuk membatalkan pengisian data pengguna baru
6.	Simpan	*Button*	Tombol untuk menyimpan data pengguna baru yang telah diisikan oleh *admin*

5.2.4.3 Perancangan Halaman *Edit Data*



Gambar 5.7 Halaman *Edit Data*

Pada Gambar 5.7 menggambarkan perancangan *interface* halaman *Edit Data*, penjelasan lebih rinci akan dijelaskan pada Tabel 5.6:

Tabel 5.6 Penjelasan halaman *Edit Data*

No	Nama Objek	Tipe	Keterangan
1.	*Navigation Menu*	*Button*	Tombol untuk navigasi menu pada aplikasi
2.	*Edit Data*	*Text View*	Teks yang menerangkan halaman yang sedang aktif digunakan pengguna
3.	*Field Data*	*Data Form*	Form untuk mengisi dan mengubah data pengguna yang telah terdaftar
4.	Tutup	*Button*	Tombol untuk menutup form edit data
5.	Batal	*Button*	Tombol untuk membatalkan pengisian form edit data
6.	Simpan	*Button*	Tombol untuk menyimpan form edit data yang telah diisikan oleh *admin*

5.2.5 Perancangan Halaman *Maps*

Gambar 5.8 Halaman *Maps*

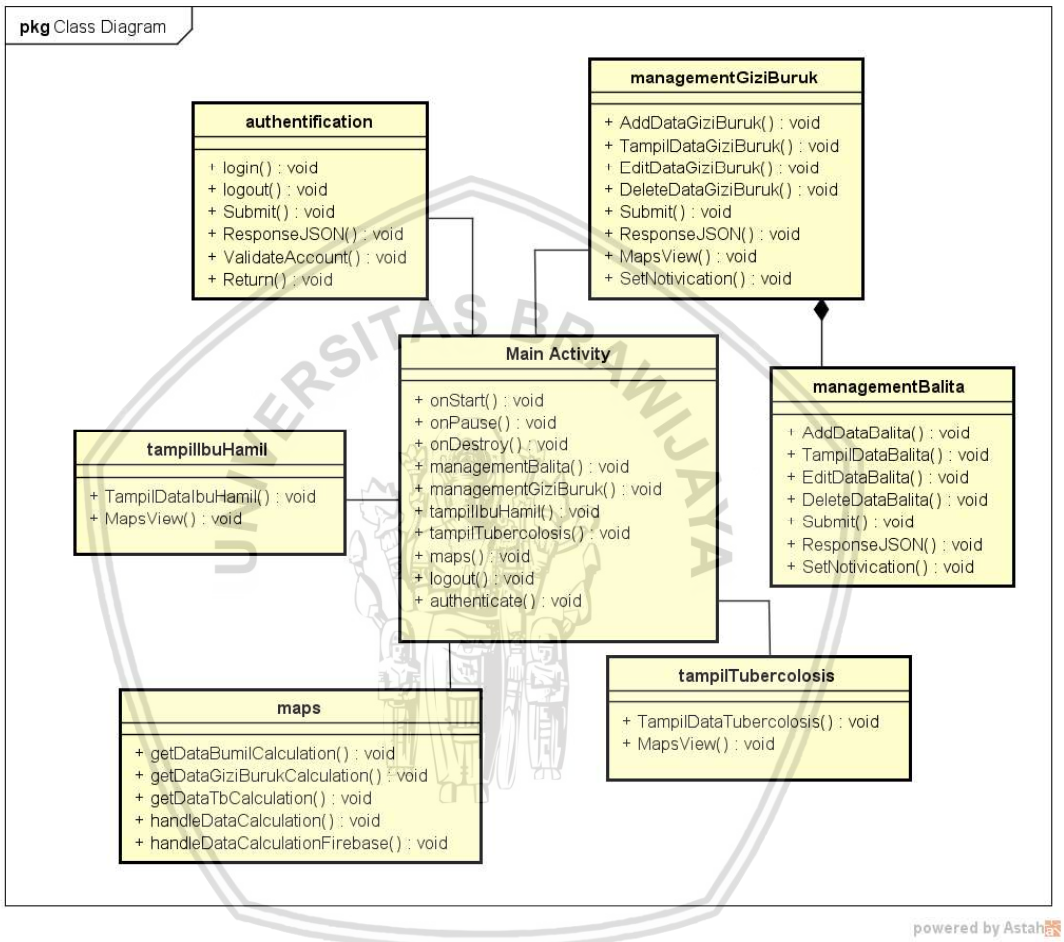
Pada Gambar 5.8 menggambarkan perancangan *interface* halaman *Maps*, penjelasan lebih rinci akan dijelaskan pada Tabel 5.7:

Tabel 5.7 Penjelasan halaman *Maps*

No	Nama Objek	Tipe	Keterangan
1.	*Navigation Menu*	*Button*	Tombol untuk navigasi menu pada aplikasi
2.	*Maps*	*Text View*	Teks yang menerangkan halaman yang sedang aktif digunakan pengguna
3.	*Maps Filter*	*Form*	Form untuk memilih data apa yang akan ditampilkan pada *Maps*
5.	Simpan	*Button*	Tombol untuk menyimpan *Maps* yang telah di filter sebelumnya
6.	Tampil	*Button*	Tombol untuk menampilkan *Maps* yang telah di filter sebelumnya
7.	*Maps*	*Maps View*	Tampilan *Maps* yang ditunjukkan kepada pengguna yang telah difilter sebelumnya

5.3 Perancangan *Class Diagram*

Pada perancangan *Class Diagram* aplikasi pelayanan kesehatan masyarakat akan menjelaskan tentang struktur dan deskripsi C*lass* yang berisi *Atribut* dan *Method Class,* serta hubungan antar *Class* untuk membangun aplikasi. Berikut merupakan perancangan *Class Diagram* yang akan digambarkan pada Gambar 5.9 dibawah ini.



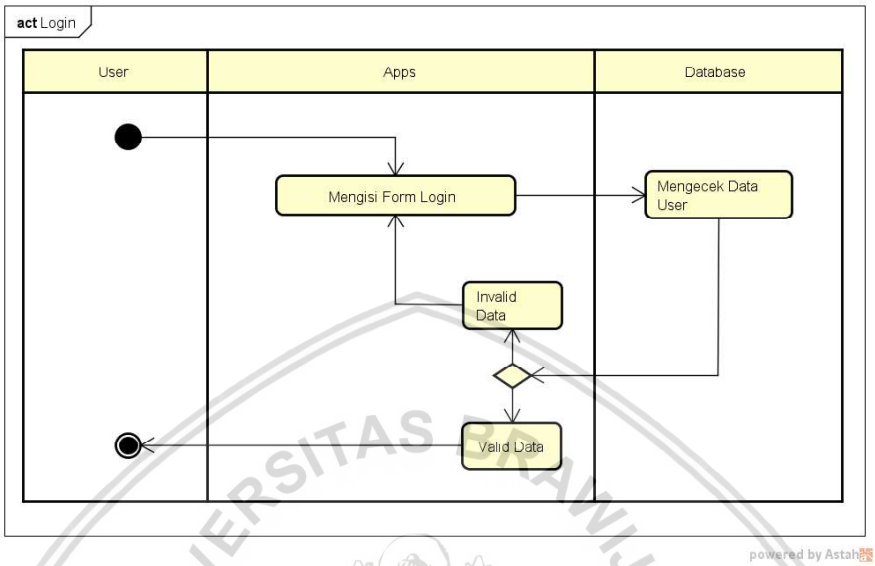
Gambar 5.9 Perancangan Class Diagram

5.4 Perancangan *Activity Diagram*

Pada perancangan *Activity Diagram* aplikasi pelayanan kesehatan masyarakat akan menjelaskan tentang aktifitas-aktifitas yang terjadi antara aplikasi dan pengguna. Hal ini akan digambarkan dengan rangkaian aktifitas yang saling berhubungan untuk melakukan tujuan tertentu. Berikut merupakan penjelasan pada perancangan *Activity Diagram.*

5.4.1 Perancangan *Activity Diagram Login*

Perancangan *Activity Diagram Login* akan digambarkan pada Gambar 5.10 dibawah ini.

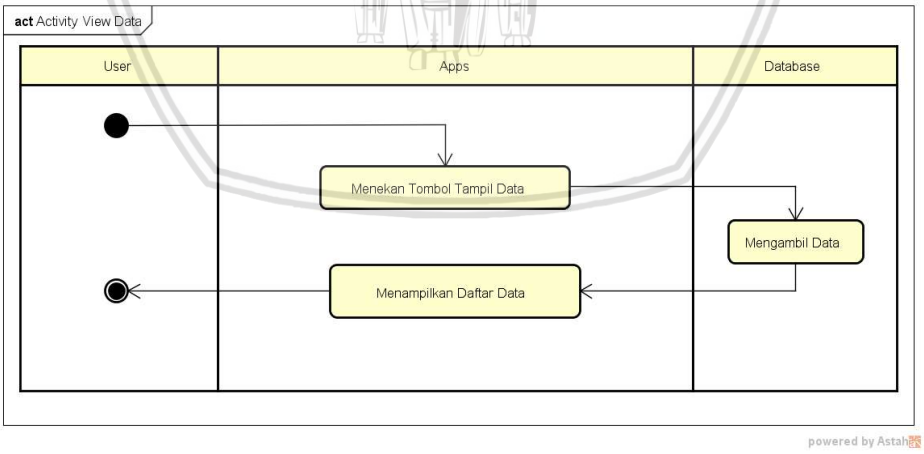


Gambar 5.10 Perancangan *Activity Diagram Login*

5.4.2 Perancangan *Activity Diagram Management Data*

5.4.2.1 Perancangan *Activity Diagram* Tampil Data

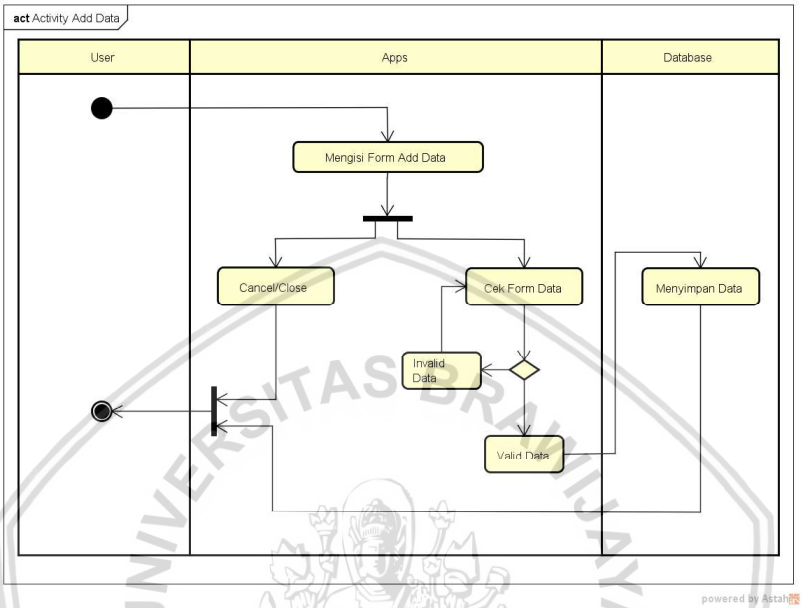
Perancangan *Activity Diagram* Tampil Data akan digambarkan pada Gambar 5.11 dibawah ini.



Gambar 5.11 Perancangan *Activity Diagram* Tampil *Data*

5.4.2.2 Perancangan *Activity Diagram Add Data*

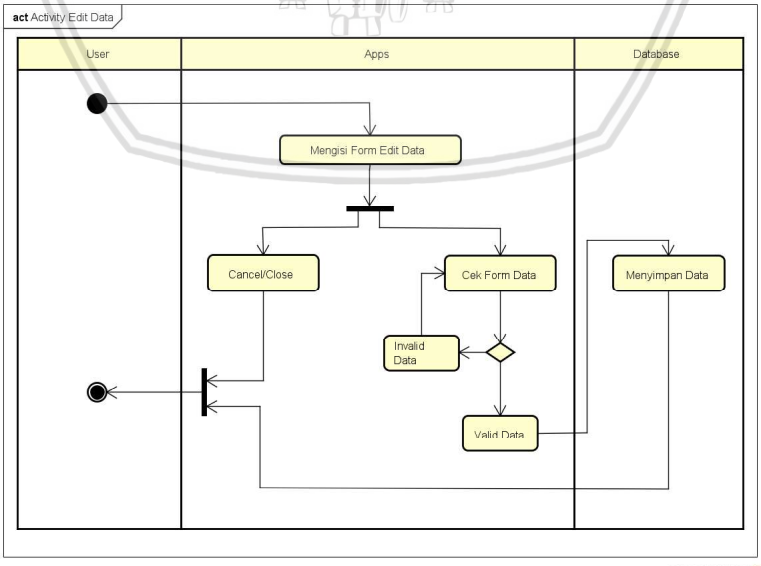
Perancangan *Activity Diagram Add Data* akan digambarkan pada Gambar 5.12 dibawah ini.



Gambar 5.12 Perancangan *Activity Diagram Add Data*

5.4.2.3 *Perancangan Activity Diagram Edit* Data

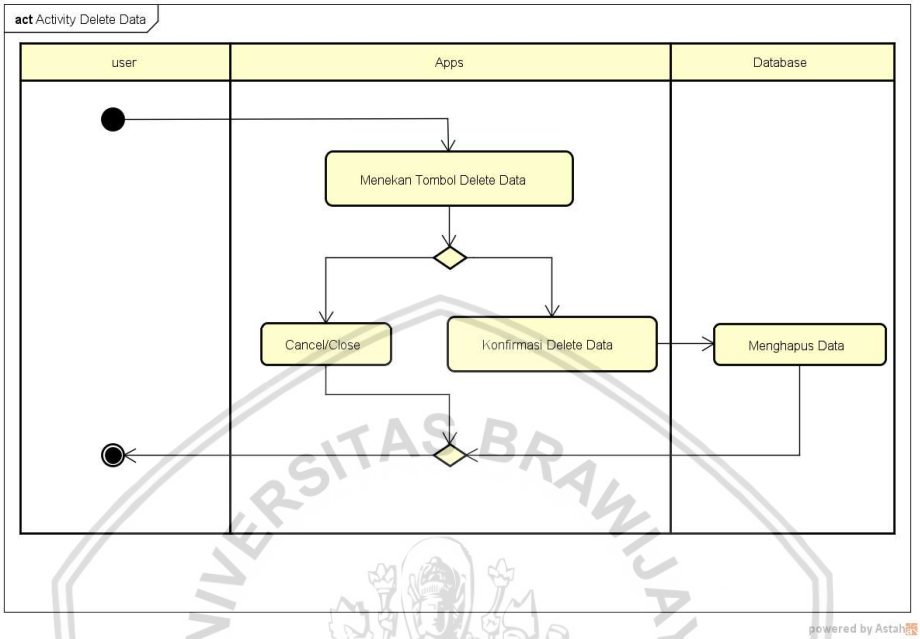
Perancangan *Activity Diagram Edit Data* akan digambarkan pada Gambar 5.13 dibawah ini.



Gambar 5.13 Perancangan *Activity Diagram Edit Data*

5.4.2.4 *Perancangan Activity Diagram Delete Data*

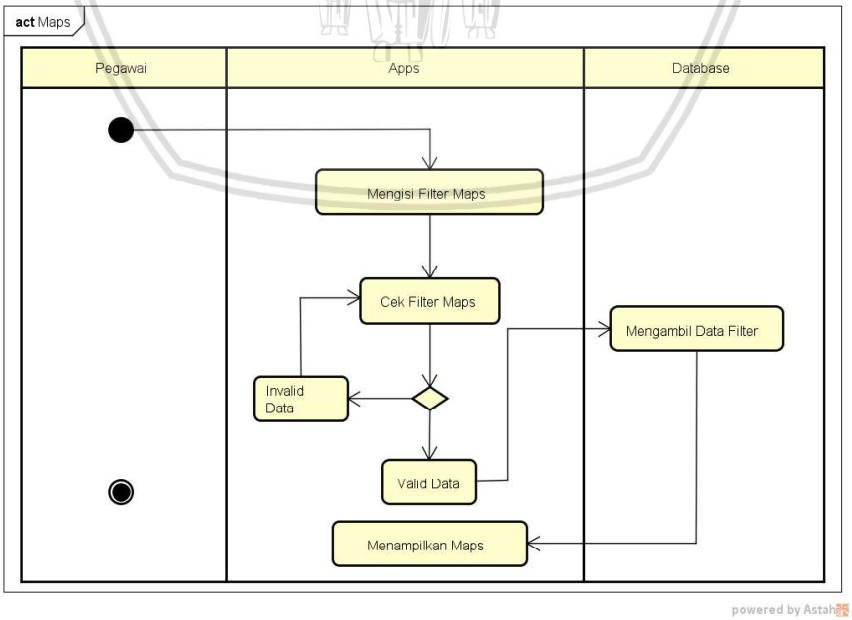
Perancangan *Activity Diagram Delete Data* akan digambarkan pada Gambar 5.14 dibawah ini.



Gambar 5.14 Perancangan *Activity Diagram Delete User*

5.4.3 Perancangan *Activity Diagram Maps*

Perancangan *Activity Diagram* Tampil *Maps* akan digambarkan pada Gambar 5.15 dibawah ini.

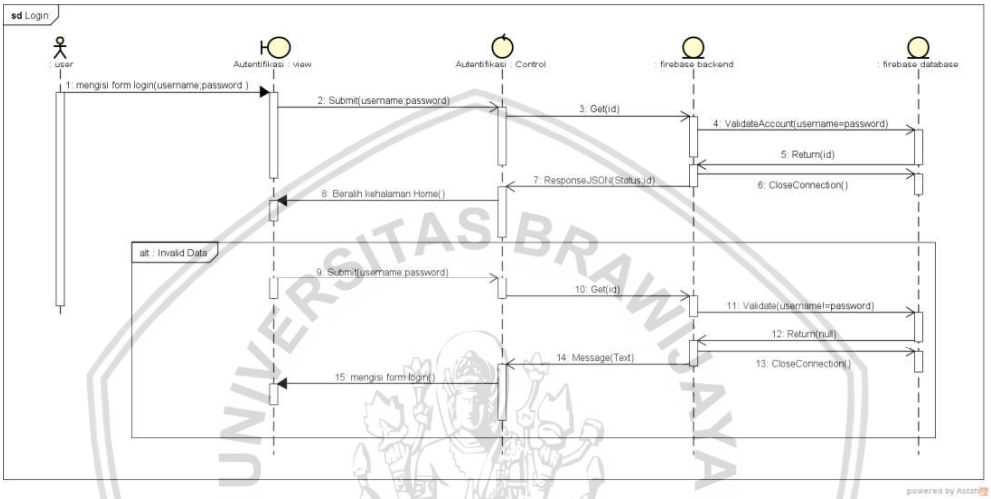


Gambar 5.15 Perancangan *Activity Diagram* Tampil *Maps*

Pada perancangan *Sequence Diagram* aplikasi pelayanan kesehatan masyarakat akan menjelaskan tentang kolaborasi peran secara dinamis antar objek. Hal ini ditunjukkan dengan rangkaian pesan yang dikirim oleh suatu objek kepada objek lainnya untuk membentuk suatu rangkaian proses. Berikut merupakan penjelasan pada perancangan *Sequence Diagram.*

5.4.4 Perancangan *Sequence Diagram Login*

Perancangan *Sequence Diagram Login* akan digambarkan pada Gambar 5.16 dibawah ini.

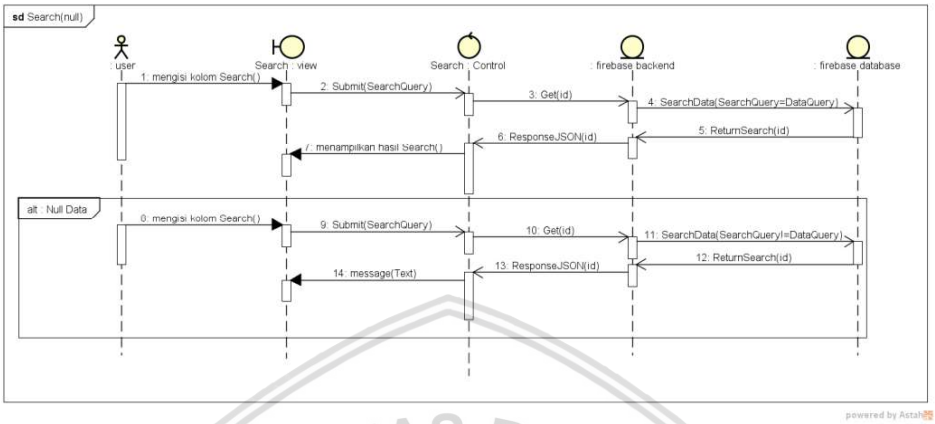


Gambar 5.16 Perancangan *Sequence Diagram Login*

Pada ***Sequence Diagram Login*** diatas ***User*** mengisi ***Form Login*** yang berisi parameter ***Username*** dan ***Password*** dan kemudian menekan tombol ***Login*** pada ***Autentifikasi : View***. Kemudian pada ***Autentifikasi : Control*** akan mendapatkan layanan ***Get*** dengan parameter ***id*** pada ***Firebase Backend*** dan memvalidasi akun, jika ***Username*** dan ***Password*** tidak sesuai maka ***Autentifikasi : Control*** mengirimkan ***Text*** peringatan bahwa ***Data*** yang dimasukkan salah. Jika ***Data*** yang dimasukkan sesuai, maka ***Firebase Database*** akan mengembalikan ***Status*** bahwa ***Data*** benar dan akan beralih pada halaman ***Home.***

5.4.5 Perancangan *Sequence Search*

Perancangan *Sequence Diagram Search* akan digambarkan pada Gambar 5.17 dibawah ini.



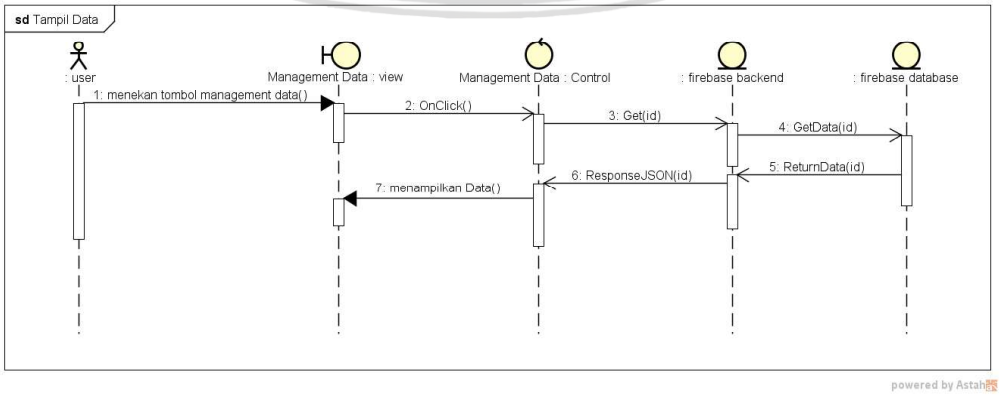
Gambar 5.17 Perancangan *Sequence Search*

Pada ***Sequence Diagram Search*** diatas, ***User*** mengisi kolom ***Search*** dan melakukan ***Submit*** dengan ***parameter SearchQuery*** dan ***Search : Control*** akan melakukan ***Get*** dengan ***parameter id*** kepada ***Firebase Backend***. Kemudian ***Firebase Backend*** mencocokkan ***SearchData*** yang ber-***parameter SearchQuery*** dan ***DataQuery*** dan akan me-***returnSearch*** dengan ***parameter id***. ***Firebase Backend*** akan mengirimkan ***ResponseJSON*** dengan ***parameter id*** kepada ***Search : Control*** dan akan ditampilkan oleh ***Search : View***. Jika ***SearchQuery*** tidak ditemukan atau ***null*** maka ***Search : Control*** akan menampilkan pesan bahwa pencarian ***data*** tidak ditemukan.

5.4.6 Perancangan *Sequence Diagram Management Data*

5.4.6.1 *Perancangan Sequence Diagram* Tampil Data

Perancangan *Sequence Diagram* Tampil Data akan digambarkan pada Gambar 5.18 dibawah ini.

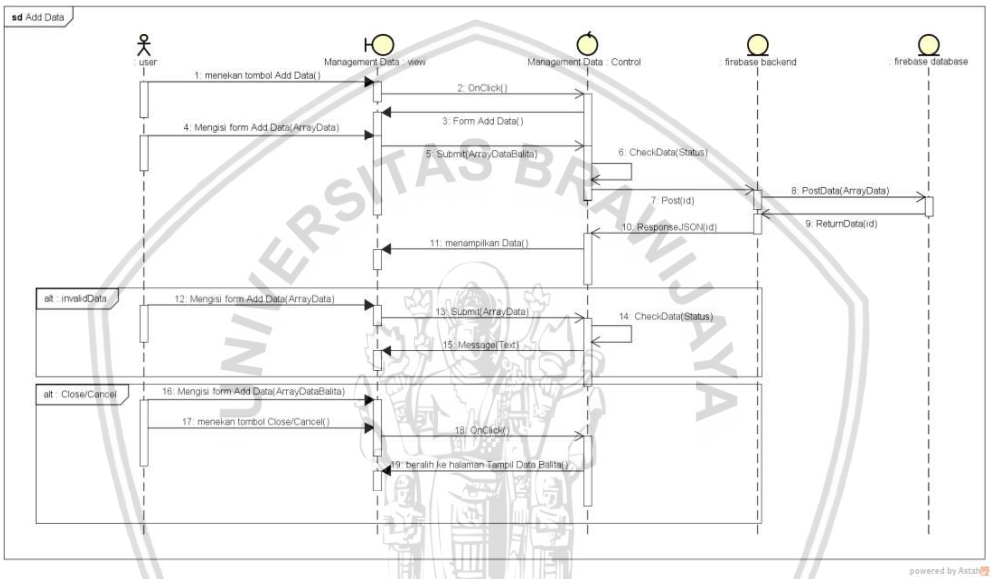


Gambar 5.18 Perancangan *Sequence Diagram* Tampil Data

Pada ***Sequence Diagram* Tampil Data**, ***User*** menekan tombol ***Management Data*** dan ***Management Data : View*** akan memproses kepada ***Management Data : Control***. Dilanjutkan dengan mendapatkan layanan ***Get*** dengan parameter ***id*** yang tersedia pada ***Firebase Backend*** dan menjalankan ***GetData*** dengan parameter ***id*** pada ***Firebase Database*** dan ***Firebase Backend*** merespon dengan ***ResponseJSON*** berparameter ***id***. Kemudian ***Management Data : Control*** akan menampilkan kepada ***Management Data : View.***

5.4.6.2 *Perancangan Sequence Diagram Add Data*

Perancangan *Sequence Diagram Add Data* akan digambarkan pada Gambar 5.19 dibawah ini.

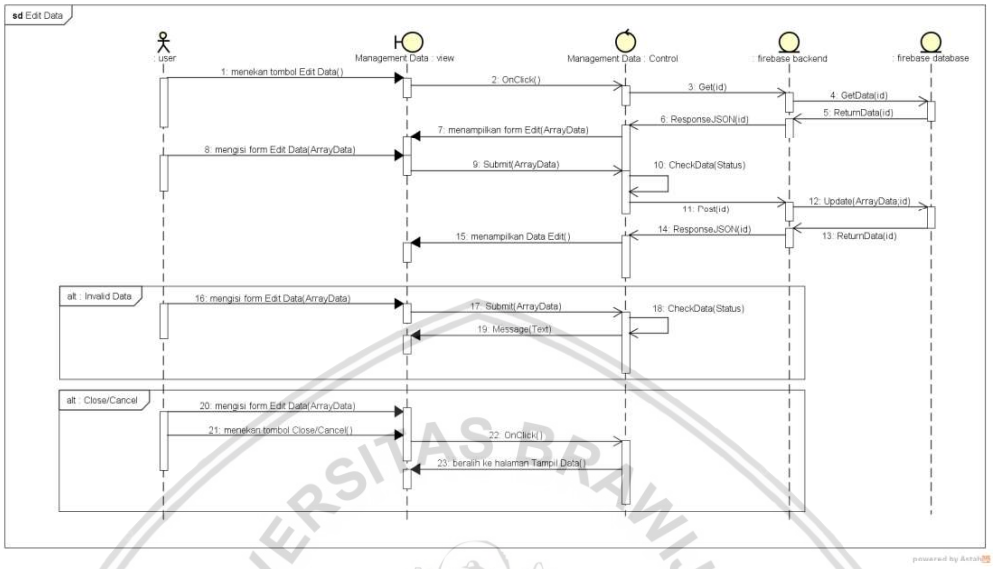


Gambar 5.19 Perancangan *Sequence Diagram Add Data*

Pada ***Sequence Diagram Add Data***, ***User*** menekan tombol ***Add Data*** dan mendapatkan ***Form*** dari ***Managemen Data : Control***. Kemudian ***User*** mengisi ***Form Add Data*** dan menekan tombol ***Submit*** yang berisi ***ArrayData***. Selanjutnya akan dilakukan ***Validasi***, jika ***Data*** inputan benar maka ***Management Data : Control*** akan melakukan ***Post*** dengan parameter ***id*** kepada ***Firebase Backend***. Kemudian melakukan ***PostData*** dengan parameter ***ArrayData*** ke dalam ***Firebase Database***. Setelah itu me-***ReturnData*** kepada ***Firebase Backend*** dan ***Management Data : Control*** akan menampilkan ***ResponseJSON*** yang didapat. Jika inputan ***Data*** salah maka pesan peringatan akan ditampilkan oleh ***Management Data : Control***. Jika ***User*** menekan tombol ***Close*** atau ***Cancel Management Data : Control*** akan mengalihkan ke halaman ***Tampil Data***.

5.4.6.3 *Perancangan Sequence Diagram Edit Data*

Perancangan *Sequence Diagram Edit Data* akan digambarkan pada Gambar 5.20 dibawah ini.

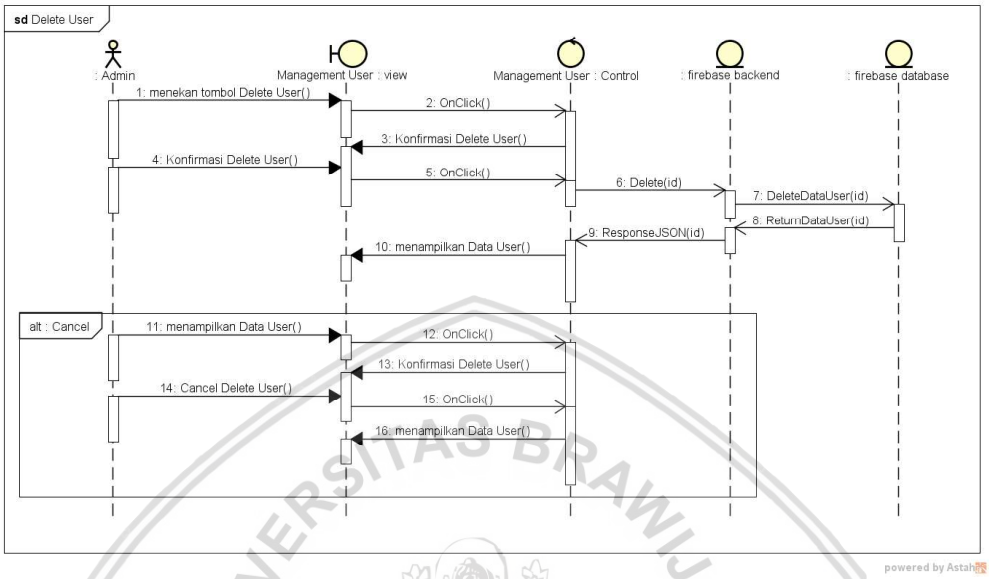


Gambar 5.20 Perancangan *Sequence Diagram Edit Data*

Pada ***Sequence Diagram Edit Data*** diatas, ***User*** akan menekan tombol ***Edit Data*** dan akan mendapatkan ***Form Edit Data*** yang telah terdaftar pada ***Firebase Database***. Dengan melakukan ***GetData*** yang ber-parameter ***id*** dan me-***ReturnData*** dan ***ResponseJSON*** yang akan ditampilkan pada ***Form Edit Data.*** Kemudian ***User*** akan mengisi ***Form Edit Data*** dan melakukan ***Submit*** yang berisi ***ArrayData*** kepada ***Management Data : Control*** dan akan dilakukan ***Validasi***. Jika ***Data*** inputan benar maka akan dilakukan ***Post*** dengan parameter ***id*** kepada layanan ***Firebase Backend*** dan melakukan ***Update*** yang berisi ***ArrayData*** dan ***id***. Kemudian akan dikembalikan dengan ***ResponseJSON*** dan akan ditampilkan oleh ***Management Data : Control*** kepada ***Management Data : View***. Jika ***Data*** inputan salah maka pesan peringatan akan ditampilkan oleh ***Management Data : Control***, jika ***User*** menekan tombol ***Close*** atau ***Cancel*** maka akan beralih ke halaman ***Tampil Data***.

5.4.6.4 *Perancangan Sequence Diagram Delete Data*

Perancangan *Sequence Diagram Delete Data* akan digambarkan pada Gambar 5.21 dibawah ini.

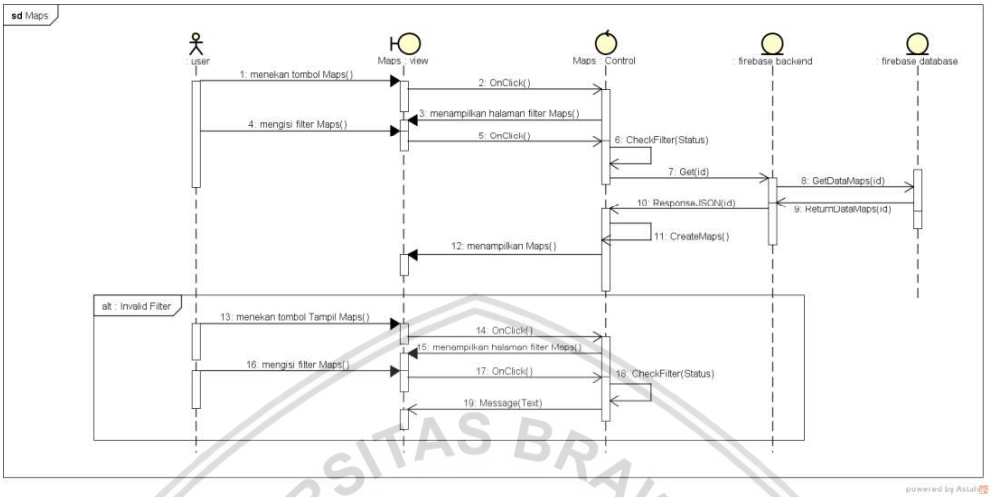


Gambar 5.21 Perancangan *Sequence Diagram Delete Data*

Pada ***Sequence Diagram Delete Data*** diatas, ***User*** akan menekan tombol ***Delete Data*** dan ***Management Data : Control*** akan melakukan ***konfirmasi Delete Data***. Jika ***User*** mengkonfirmasi penghapusan, maka ***Management Data*** melakukan layanan ***Delete*** pada ***Firebase Backend*** yang berisi parameter ***id***. Kemudian ***Firebase Backend*** melakukan ***DeleteData*** dengan parameter ***id*** yang ada pada ***Firebase Database***. Kemudian me-***ReturnData*** dengan melakukan ***ResponseJSON*** kepada ***Management Data : Control*** dan menampilkan ***Data*** kepada ***Management Data : View***. Jika ***User*** menekan tombol ***Cancel***, maka ***Management Data : Control*** akan menampilkan ***Tampil Data***.

5.4.7 Perancangan Sequence Diagram Maps

Perancangan *Sequence Diagram Maps* akan digambarkan pada Gambar 5.22 dibawah ini.

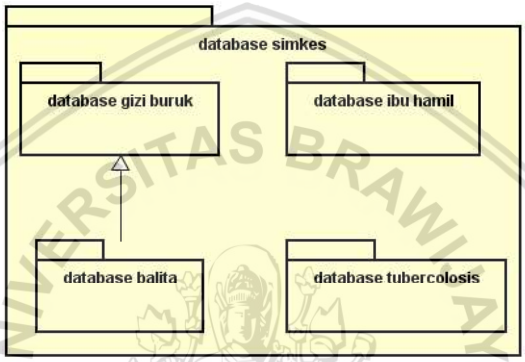


Gambar 5.22 Perancangan *Sequence Diagram* Tampil *Maps*

Pada *Sequence Diagram* tampil *Maps* diatas, ***Admin*** menekan tombol tampil *maps* kemudian ***Maps : View*** akan menampilkan filter untuk tampilan *maps*. Setelah filter diisi maka ***Maps : Control*** akan mengecek filter, jika sudah sesuai maka akan melakukan ***GET*** data kepada ***Firebase Backend*** dan ***Firebase Database***. Setelah itu akan dikembalikan dalam bentuk ***ResponseJSON*** dan meng-*create maps* dan menampilkannya pada ***Maps : View***. Jika filter tidak sesuai maka ***Maps : Control*** akan menampilkan pesan teks jika pengisian filter tidak sesuai.

5.5 Perancangan *Database*

Perancangan *database* dilaksanakan dengan menggunakan *JSON scheme* yang dirancang dengan menuliskan format data *JSON* dengan *title, type, properties* sebagai atribut properti yang ada pada objek. Pada penelitian ini menggunakan *firebase realtime database* sebagai simulasi penyimpanan data yang memiliki empat *database* yaitu *database* balita, gizi buruk, ibu hamil, dan *tubercolosis.* Dimana *database* balita merupakan bagian dari *database* gizi buruk. Penggunaan *firebase database* pada aplikasi untuk membatasi kesalahan informasi data kesehatan pada *server online* pada aplikasi informasi Kesehatan dalam proses pengembangan sistem pelayanan kesehatan. Perancangan *database* akan ditunjukkan pada Gambar 5.23 dibawah ini.



Gambar 5.23 Perancangan Struktur *Database*

5.5.1 Perancangan *Database* Balita

Perancangan data balita akan digambarkan pada gambar 5.24

```
{
  "databalita": {
    "type": "object",
    "properties": {
      "idbalita": {
        "type": "string"
      },
      "idjeniskelamin": {
        "type": "string"
      },
      "jeniskelamin": {
        "type": "string"
      },
      "namaBalita": {
        "type": "string"
      },
      "namaOrangTua": {
        "type": "string"
      }
    }
  }
}
```

Gambar 5.24 Perancangan *database* balita

5.5.2 Perancangan *Database* Gizi Buruk

Perancangan data gizi buruk akan digambarkan pada gambar 5.25

```
{
  "datagiziburuk": {
    "type": "object",
    "properties": {
      "alamat": {
        "type": "string"
      },
      "bb_kg": {
        "type": "string"
      },
      "bulan_keluar": {
        "type": "string"
      },
      "bulan_masuk": {
        "type": "string"
      },
      "idapp_user": {
        "type": "string"
      },
      "idbalita": {
        "type": "string"
      },
      "iddata_gizi_buruk": {
        "type": "string"
      },
      "idjeniskelamin": {
        "type": "string"
      },
      "idkecamatan": {
        "type": "string"
      },
      "idkelurahan": {
        "type": "string"
      },
      "idkota": {
        "type": "string"
      },
      "idpropinsi": {
        "type": "string"
      },
      "idpuskesmas": {
        "type": "string"
      },
      "kecamatan": {
        "type": "string"
      },
      "kelurahan": {
        "type": "string"
      },
      "kota": {
        "type": "string"
      },
      "nama_balita": {
        "type": "string"
      },
      "nama_orang_tua": {
        "type": "string"
      },
      "no_kelurahan": {
        "type": "string"
      },
      "no_puskesmas": {
        "type": "string"
      },
      "propinsi": {
        "type": "string"
      },
      "puskesmas": {
        "type": "string"
      },
      "rt": {
        "type": "string"
      },
      "rw": {
        "type": "string"
      },
      "status": {
        "type": "string"
      },
      "status_keluarga_gakin": {
        "type": "string"
      },
      "status_keluarga_nongakin": {
        "type": "string"
      },
      "systemtime": {
        "type": "string"
      },
      "tanggal": {
        "type": "string"
      },
      "tb_cm": {
        "type": "string"
      },
      "umur_bulan": {
        "type": "string"
      },
      "username": {
        "type": "string"
      }
    }
  }
}
```


Gambar 5.25 Perancangan *database* gizi buruk

5.5.3 Perancangan *Database* Ibu Hamil

Perancangan data ibu hamil akan digambarkan pada gambar 5.26

```
{
  "databumil": {
    "type": "object",
    "properties": {
      "alamat": {
        "type": "string"
      },
      "anc_terpadu_albumin": {
        "type": "string"
      },
      "anc_terpadu_gds": {
        "type": "string"
      },
      "anc_terpadu_hbsag": {
        "type": "string"
      },
      "anc_terpadu_hiv": {
        "type": "string"
      },
      "anc_terpadu_reduksi": {
        "type": "string"
      },
      "bb": {
        "type": "string"
      },
      "faktor_resiko": {
        "type": "string"
      },
      "hamil_ke": {
        "type": "string"
      },
      "hb_golda": {
        "type": "string"
      },
      "hpht": {
        "type": "string"
      },
      "iddata_bumil": {
        "type": "string"
      },
      "imunisasi_tt_pemberian_imunisasi_tt": {
        "type": "string"
      },
      "imunisasi_tt_status_imunisasi_tt": {
        "type": "string"
      },
      "jarak_kehamilan": {
        "type": "string"
      },
      "kehamilan_minggu": {
        "type": "string"
      },
      "kelurahan": {
        "type": "string"
      },
      "keterangan_bpjs": {
        "type": "string"
      },
      "lila_imt": {
        "type": "string"
      },
      "nama_ibu": {
        "type": "string"
      },
      "nama_suami": {
        "type": "string"
      },
      "no_nik_ibu": {
        "type": "string"
      },
      "pendeteksi_resiko_masyarakat": {
        "type": "string"
      },
      "pendeteksi_resiko_nakes": {
        "type": "string"
      },
      "puskesmas": {
        "type": "string"
      },
      "rencana_persalinan_biaya": {
        "type": "string"
      },
      "rencana_persalinan_kendaraan": {
        "type": "string"
      },
      "rencana_persalinan_tempat": {
        "type": "string"
      },
      "rt": {
        "type": "string"
      },
      "rw": {
        "type": "string"
      },
      "tafsiran_persalinan": {
        "type": "string"
      },
      "tanggal": {
        "type": "string"
      },
      "tb": {
        "type": "string"
      },
      "tensi": {
        "type": "string"
      },
      "umur_ibu": {
        "type": "string"
      }
    }
  }
}
```


Gambar 5.26 Perancangan *database* ibu hamil

5.5.4 Perancangan *Database Tubercolosis*

Perancangan data *tubercolosis* akan digambarkan pada gambar 5.27

Gambar 5.27 Perancangan *database tubercolosis*

BAB 6 IMPLEMENTASI

Bab ini membahas tentang implementasi aplikasi berdasarkan analisis kebutuhan dan proses perancangan aplikasi yang telah dibuat. Pembahasan pada bab ini terdiri dari spesifikasi, batasan dan impelementasi metode pembuatan aplikasi informasi kesehatan masyarakat kota malang berbasis *android mobile native*.

6.1 Spesifikasi

Pada sub bab spesifikasi ini, membahas tentang bagaimana spesifikasi perangkat keras dan perangkat lunak yang digunakan dalam membangun aplikasi informasi kesehatan masyarakat kota malang berbasis *mobile native android* ini.

6.1.1 Spesifikasi Perangkat Keras

Penjelasan spesifikasi perangkat keras yang digunakan dalam pembuatan aplikasi informasi kesehatan masyarakat kota malang berbasis *mobile native android* ini dijelaskan dalam Tabel 6.1 sebagai berikut:

Tabel 6.1 Spesifikasi Perangkat Keras

No.	Jenis	Spesifikasi
1.	Processor	Intel (R) Core(TM) i5-5200U CPU @2.20GHz
2.	Installed Memory (RAM)	8 GB
3.	System Type	64-*bit* Operating System

6.1.2 Spesifikasi Perangkat Lunak

Penjelasan spesifikasi perangkat lunak yang digunakan dalam pembuatan aplikasi informasi kesehatan masyarakat kota malang berbasis *mobile native android* ini dijelaskan dalam Tabel 6.2 sebagai berikut:

Tabel 6.2 Spesifikasi Perangkat Lunak

No.	Jenis	Spesifikasi
1.	Aplikasi operasi	Windows 10 Ultimate
2.	Bahasa pemrograman	Java
3.	Tools pemrograman	Android Studio

6.2 Batasan Implementasi

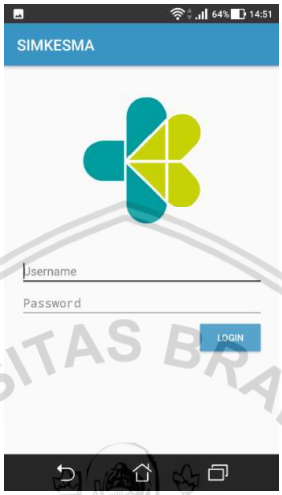
Penjelasan tentang batasan implementasi yang terdapat didalam skripsi ini adalah sebagai berikut:

1. Data yang digunakan berasal dari *Database* SIMKES Kota Malang.
2. Hasil keluaran program dalam bentuk aplikasi *mobile*.
3. Menggunakan bahasa pemrograman *Java* untuk mengembangkan implementasi pada skripsi ini.
4. Menggunakan *tools* pemrograman *Android Studio*.

6.3 Implementasi *Interface*

Pada implementasi *interface* disini merupakan hasil implementasi elemen antarmuka yang telah dirancang sebelumnya dan dilakukan beberapa penyesuaian dengan perangkat mobile.

6.3.1 *Interface* Login



Gambar 6.1 *Interface Login*

Pada Gambar 6.1 merupakan implementasi halaman login pada *interface* aplikasi informasi kesehatan masyarakat kota Malang berbasis *mobile native android.*

6.3.2 *Interface Home*

Gambar 6.2 *Interface Home*

Pada Gambar 6.2 merupakan implementasi halaman home pada *interface* aplikasi informasi kesehatan masyarakat kota Malang berbasis *mobile native android*.

6.3.3 *Interface Management* Data Ibu Hamil

Gambar 6.3 *Interface Management* Data Ibu Hamil

Pada Gambar 6.3 merupakan implementasi halaman *Management* data Ibu Hamil pada *interface* aplikasi informasi kesehatan masyarakat Kota Malang berbasis *mobile native android*.

6.3.3.1 *Interface* Tampil Data Ibu Hamil

Gambar 6.4 *Interface* Tampil Data Ibu Hamil

Pada Gambar 6.4 merupakan implementasi halaman tampil data Ibu Hamil pada *interface* aplikasi informasi kesehatan masyarakat Kota Malang berbasis *mobile native android*.

6.3.4 *Interface* Management Data Gizi Buruk

Gambar 6.5 *Interface Management* Data Gizi Buruk

Pada Gambar 6.5 merupakan implementasi halaman *Management* Data Gizi Buruk pada *interface* aplikasi informasi kesehatan masyarakat Kota Malang berbasis *mobile native android*.

6.3.4.1 *Interface* Tampil Data Gizi Buruk

Gambar 6.6 *Interface* Tampil Data Gizi Buruk

Pada Gambar 6.6 merupakan implementasi halaman tampil data Gizi Buruk pada *interface* aplikasi informasi kesehatan masyarakat kota Malang berbasis *mobile native android*.

6.3.4.2 *Interface Add* dan *Edit* Data Gizi Buruk

Gambar 6.7 *Interface Add* dan *Edit* Data Gizi Buruk

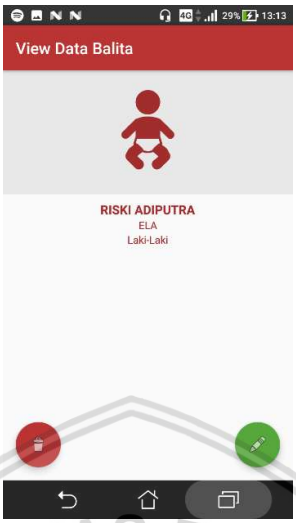
Pada Gambar 6.7 merupakan implementasi halaman *add* dan *edit* data Gizi Buruk pada *interface* aplikasi informasi kesehatan masyarakat Kota Malang berbasis *mobile native android*.

6.3.5 *Interface* Management Data Balita

Gambar 6.8 *Interface Management* Data Balita

Pada Gambar 6.8 merupakan implementasi halaman *Management* Data Balita pada *interface* aplikasi informasi kesehatan masyarakat Kota Malang berbasis *mobile native android*.

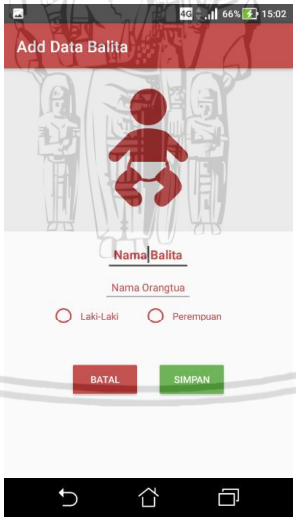
6.3.5.1 *Interface Tampil Data* Balita



Gambar 6.9 *Interface* Tampil Data Balita

Pada Gambar 6.9 merupakan implementasi halaman tampil data Balita pada *interface* aplikasi informasi kesehatan masyarakat kota Malang berbasis *mobile native android*.

6.3.5.2 *Interface Add* dan *Edit* Data Balita



Gambar 6.10 *Interface Add* dan *Edit* Data Balita

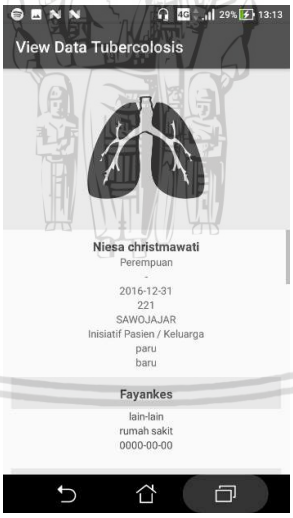
Pada Gambar 6.10 merupakan implementasi halaman *add* dan *edit* data Balita pada *interface* aplikasi informasi kesehatan masyarakat Kota Malang berbasis *mobile native android*.

6.3.6 Interface *Management* Data *Tubercolosis*

Gambar 6.11 *Interface Management* Data *Tubercolosis*

Pada Gambar 6.11 merupakan implementasi halaman *Management* Data *Tubercolosis* pada *interface* aplikasi informasi kesehatan masyarakat Kota Malang berbasis *mobile native android*.

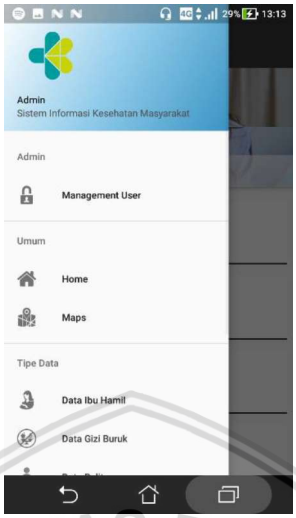
6.3.6.1 Interface Tampil Data *Tubercolosis*



Gambar 6.12 *Interface* Tampil Data *Tubercolosis*

Pada Gambar 6.12 merupakan implementasi halaman tampil data *Tubercolosis* pada *interface* aplikasi informasi kesehatan masyarakat kota Malang berbasis *mobile native android*.

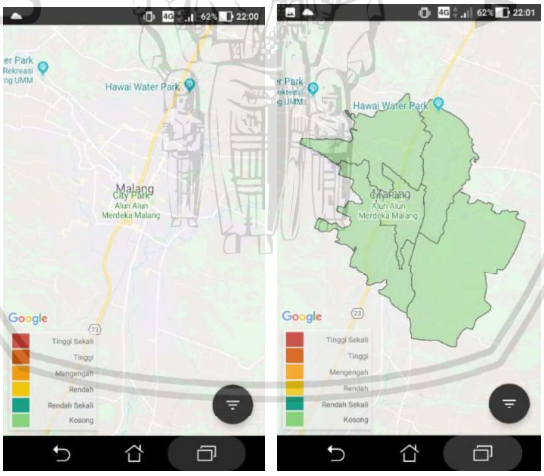
6.3.7 *Interface Navigation Menu*



Gambar 6.13 *Interface Navigation Menu*

Pada Gambar 6.13 merupakan implementasi halaman *Navigation Menu* pada *interface* aplikasi informasi kesehatan masyarakat kota Malang berbasis *mobile native android*.

6.3.8 *Interface Maps*



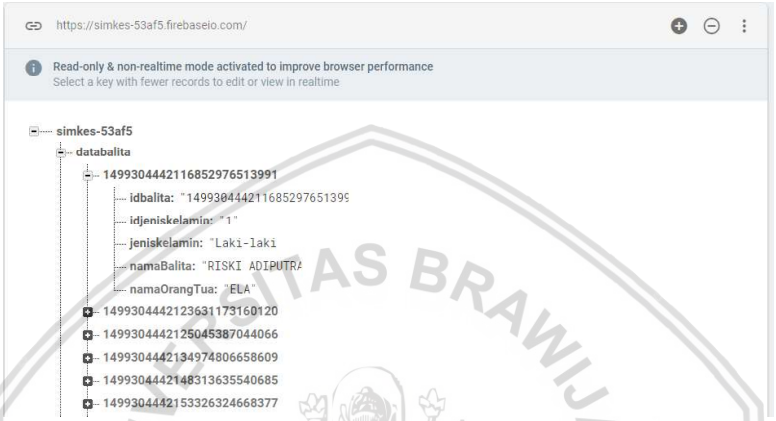
Gambar 6.14 *Interface Maps*

Pada Gambar 6.14 merupakan implementasi halaman *Maps* pada *interface* aplikasi informasi kesehatan masyarakat kota Malang berbasis *mobile native android*.

6.4 Implementasi *Database*

Pada implementasi *database,* berikut merupakan hasil dari perancangan *database* yang sebelumnya telah dilakukan dan menggunakan *Google firebase realtime database* sehingga dapat menjadi rangkaian database yang lengkap. Rangkaian *database* tersebut dapat melakukan fungsi utama seperti *tampil*, *add*, *edit*, dan *delete* data serta digunakan untuk kebutuhan pada *maps.*

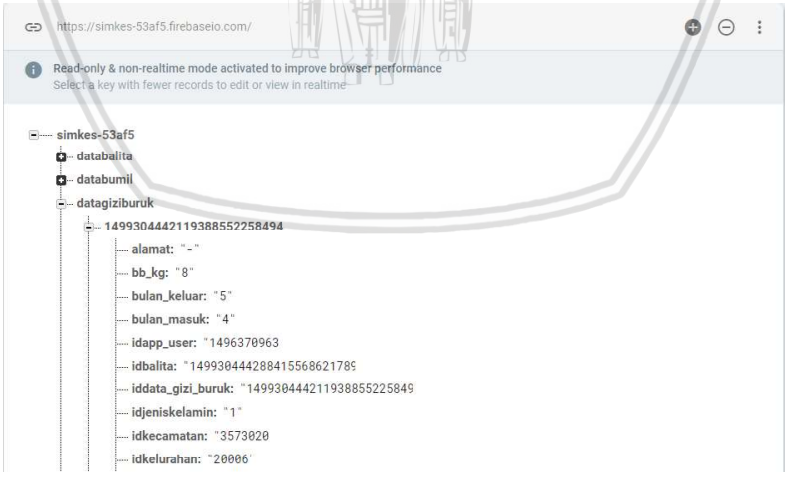
6.4.1 Implementasi *Database* Balita



Gambar 6.15 Implementasi *Database* Balita

Pada Gambar 6.15 merupakan penjelasan tentang *database* dari data balita beserta atribut data yang dimiliki pada setiap Id data balita yang sesuai dengan perancangan *database* balita sebelumnya.

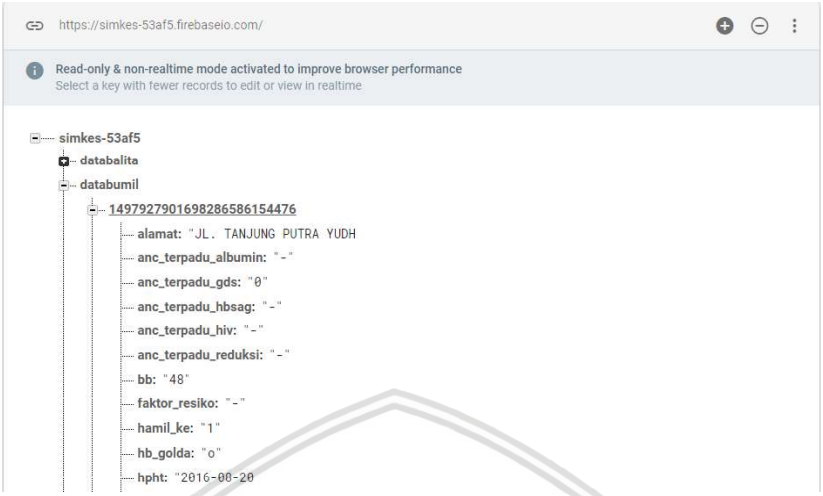
6.4.2 Implementasi *Database* Gizi Buruk



Gambar 6.16 Implementasi *Database* Gizi Buruk

Pada Gambar 6.16 merupakan penjelasan tentang *database* dari data gizi buruk beserta atribut data yang dimiliki pada setiap Id data gizi buruk yang sesuai dengan perancangan *database* gizi buruk sebelumnya.

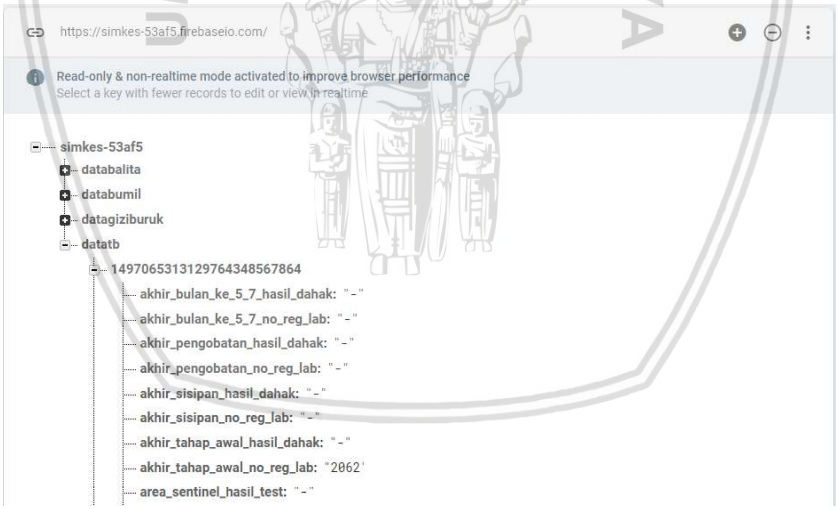
6.4.3 Implementasi *Database* Ibu Hamil



Gambar 6.17 Implementasi *Database* Ibu Hamil

Pada Gambar 6.17 merupakan penjelasan tentang *database* dari data ibu hamil beserta atribut data yang dimiliki pada setiap Id data ibu hamil yang sesuai dengan perancangan *database* ibu hamil sebelumnya.

6.4.4 Implementasi *Database Tubercolosis*



Gambar 6.18 Implementasi *Database Tubercolosis*

Pada Gambar 6.18 merupakan penjelasan tentang *database* dari data *tubercolosis* beserta atribut data yang dimiliki pada setiap Id data *tubercolosis* yang sesuai dengan perancangan *database tubercolosis* sebelumnya.

6.5 Implementasi Kode Program

6.5.1 Implementasi Kode Program Add Data

Tabel 6.3 Kode Program Add Data

No	Kode
1.	`package name;`
2.	
3.	`import library;`
4.	
5.	`public class AddData extends AppCompatActivity implements`
6.	`View.OnClickListener {`
7.	`    //variabel objek`
8.	`    Object object;`
9.	`    Button savebtn,cancelbtn;`
10.	
11.	
12.	`    //variabel database`
13.	`    private DatabaseReference mFirebaseDatabase;`
14.	`    private FirebaseDatabase mFirebaseInstance;`
15.	
16.	`    @Override`
17.	`    protected void onCreate(Bundle savedInstanceState) {`
18.	`        super.onCreate(savedInstanceState);`
19.	`        setContentView(R.layout.activity_add_data);`
20.	`        //instansiasi database`
21.	`        mFirebaseInstance = FirebaseDatabase.getInstance();`
22.	`        mFirebaseDatabase = mFirebaseInstance.getReference("data");`
23.	`        //deklarasi fungsi dan id toolbar`
24.	`        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);`
25.	`        setSupportActionBar(toolbar);`
26.	`        //deklarasi id objek`
27.	`        object = (Object) findViewById(R.id.object);`
28.	`        //instansiasi fungsi dan id button`
29.	`        cancelbtn = (Button) findViewById(R.id.cancelbtn);`
30.	`        cancelbtn.setOnClickListener(this);`
31.	`        savebtn = (Button) findViewById(R.id.savebtn);`
32.	`        savebtn.setOnClickListener(this);`
33.	`    }`
34.	`    //fungsi memilih radio button`
35.	`           public void onRadioButtonClicked(View view) {...}`
36.	
37.	`    @Override`
38.	`    public void onClick(View v) {`
39.	`        //saat memilih tombol simpan`
40.	`        if (v.getId() == R.id.savebtn) {`
41.	`            //mengubah object menjadi string`
42.	`            object = atributobject.getText().toString();`
43.	`            //mendapatkan id key pada firebase dengan format String`
44.	`            String id = mFirebaseDatabase.push().getKey();`
45.	`            //instansiasi data beserta atributnya`
46.	`            Data jenisdata = new Data (atributobject);`
47.	`            //membuat notifikasi bahwa data telah tersimpan`
48.	`            Toast.makeText(AddData.this,"Pengisian data Sukses",`
49.	`Toast.LENGTH_SHORT).show();`
50.	`            //Menyimpan data`
51.	`            mFirebaseDatabase.child(id).setValue(jenisdata);`
52.	`            finish();`

```
53.         //jika memilih tombol cancel
54.         }if (v.getId()== R.id.cancelbtn){
55.             exitByBackKey();
56.             //membuat notifikasi bahwa add dibatalkan
57.             Toast.makeText(AddData.this, "Pengisian data dibatalkan",
58. Toast.LENGTH_SHORT).show();
59.         }
60.
61.     }
62.     //fungsi untuk membuat peringatan saat ingin keluar dari pengisian
63. data
64.     public boolean onKeyDown(int keyCode, KeyEvent event) {...}
65.     protected void exitByBackKey() {...}
```

6.5.2 Implementasi Kode Program Edit Data

Tabel 6.4 Kode Program Edit Data

No	Kode

```
1.  package name;
2.
3.  import library;
4.
5.  public class EditData extends AppCompatActivity implements
6.  View.OnClickListener {
7.      //variabel objek
8.      Object object;
9.      Button savebtn,cancelbtn;
10.
11.
12.     //variabel database
13.     private DatabaseReference mFirebaseDatabase;
14.     private FirebaseDatabase mFirebaseInstance;
15.
16.     @Override
17.     protected void onCreate(Bundle savedInstanceState) {
18.         super.onCreate(savedInstanceState);
19.         setContentView(R.layout.activity_edit_data);
20.         //mengirim data activity menggunakan bundle
21.         Bundle bundle = getIntent().getExtras();
22.         //atribut bundle beserta keynya
23.         key = bundle.getString("key");
24.         //instansiasi database
25.         mFirebaseInstance = FirebaseDatabase.getInstance();
26.         mFirebaseDatabase =
27. mFirebaseInstance.getReference("data").child(id);
28.         //deklarasi fungsi dan id toolbar
29.         Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
30.         setSupportActionBar(toolbar);
31.         //deklarasi id objek
32.         object = (Object) findViewById(R.id.object);
33.         //instansiasi fungsi dan id button
34.         cancelbtn = (Button) findViewById(R.id.cancelbtn);
35.         cancelbtn.setOnClickListener(this);
36.         savebtn = (Button) findViewById(R.id.savebtn);
37.         savebtn.setOnClickListener(this);
38.         //set text data yang akan diubah
39.         Objek.setText(key);
40.     }
41.     //fungsi memilih radio button
```

```
42.     public void onRadioButtonClicked(View view) {...}
43.
44.     @Override
45.     public void onClick(View v) {
46.         //saat memilih tombol simpan
47.         if (v.getId() == R.id.savebtn) {
48.             //mengubat text menjadi String
49.             String objectedit = object.getText().toString();
50.             //mengubah object menjadi string
51.             object = atributobject.getText().toString();
52.             //untuk mengosongkan field text saat mengedit
53.             if (!TextUtils.isEmpty(objectedit)) {
54.                 object = objectedit;
55.             }
56.             //instansiasi data beserta atributnya
57.             Data jenisdata = new Data (atributobject);
58.             //membuat notifikasi bahwa data telah terupdate
59.             Toast.makeText(AddData.this,"Update data Sukses",
60. Toast.LENGTH_SHORT).show();
61.             //Menyimpan data
62.             mFirebaseDatabase.child(id).setValue(jenisdata);
63.             //beralih ke class management data
64.             Intent intent = new Intent(EditData.this,
65. ManagementData.class);
66.             startActivity(intent);
67.             finish();
68.         //jika memilih tombol cancel
69.         }if (v.getId()== R.id.cancelbtn){
70.             exitByBackKey();
71.             //membuat notifikasi bahwa update dibatalkan
72.             Toast.makeText(EditData.this, "Update data dibatalkan",
73. Toast.LENGTH_SHORT).show();
74.         }
75.
76.     }
77.     //fungsi untuk membuat peringatan saat ingin keluar dari pengisian
78. data
79.     public boolean onKeyDown(int keyCode, KeyEvent event) {...}
     protected void exitByBackKey() {...}
```

6.5.3 Implementasi Kode Program *Delete* Data

Tabel 6.5 Kode Program *Delete* Data

No	Kode
1.	`//instansiasi fungsi dan id floating action button delete`
2.	`FloatingActionButton fabDelete = (FloatingActionButton)`
3.	`findViewById(R.id.fabDelete);`
4.	`fabDelete.setOnClickListener(new View.OnClickListener() {`
5.	`@Override`
6.	`public void onClick(View view) {`
7.	`//intent kedalam view data`
8.	`Intent intent = new Intent(ViewData.this,`
9.	`ManagementData.class);`
10.	`//menghapus object yang ada di database`
11.	`String object = null;`
12.	`//instansiasi data beserta atributnya`
13.	`Data jenisdata = new Data(atributobject);`
14.	`//menyimpan data`
15.	`mFirebaseDatabase.setValue(balitadata);`

```
16.                 finish();
17.                 //memberi notifikasi
18.                 Toast.makeText(ViewBalita.this, "Data balita telah
19. dihapus", Toast.LENGTH_SHORT).show();
20.             }
21.         });
```

6.5.4 Implementasi Kode Program *Delete* Data

Tabel 6.6 Kode Program *View* Data

No	Kode
1.	`package name;`
2.	
3.	`import library;`
4.	
5.	`public class ViewBalita extends AppCompatActivity {`
6.	`    //variabel object`
7.	`    Object object;`
8.	
9.	`    //variabel database`
10.	`    private DatabaseReference mFirebaseDatabase;`
11.	`    private FirebaseDatabase mFirebaseInstance;`
12.	
13.	`    @Override`
14.	`    protected void onCreate(Bundle savedInstanceState) {`
15.	`        super.onCreate(savedInstanceState);`
16.	`        setContentView(R.layout.activity_view_data);`
17.	`        //deklarasi id object`
18.	`        namaobject = (TextView) findViewById(R.id.textBalita_data);`
19.	`        //mengirim data activity menggunakan bundle`
20.	`        Bundle bundle = getIntent().getExtras();`
21.	`        //atribut bundle beserta keynya`
22.	`        Object key = bundle.getString("key");`
23.	`        //instansiasi database`
24.	`        mFirebaseInstance = FirebaseDatabase.getInstance();`
25.	`        mFirebaseDatabase =`
26.	`mFirebaseInstance.getReference("data").child(id);`
27.	`        //mengambil konteks teks berdasarkan id`
28.	`        Toast.makeText(getApplicationContext(), id,`
29.	`                Toast.LENGTH_SHORT).show();`
30.	`        //set text data yang akan ditampilkan`
31.	`        object.setText(key);`
32.	`        //floating action button untuk edit data`
33.	`        FloatingActionButton fabEdit = (FloatingActionButton)`
34.	`findViewById(R.id.fabEdit);`
35.	`        fabEdit.setOnClickListener(new View.OnClickListener() {...});`
36.	`        //floating action button untuk delete data`
37.	`        FloatingActionButton fabDelete = (FloatingActionButton)`
38.	`findViewById(R.id.fabDelete);`
39.	`        fabDelete.setOnClickListener(new View.OnClickListener()`
40.	`{...});`
41.	`    }`
42.	`}`

6.5.5 Implementasi Kode Program *Maps*

Tabel 6.7 Kode Program *Maps*

No	Kode
1.	package name;
2.	
3.	import library;
4.	
5.	public class Maps extends AppCompatActivity
6.	implements NavigationView.OnNavigationItemSelectedListener,
7.	OnMapReadyCallback {
8.	
9.	// Konstanta Id untuk memanggil FilterActivity
10.	private static final int FILTER_REQUEST_KEY = 998;
11.	
12.	private FloatingActionButton filterButton;
13.	private FrameLayout helperIndicator;
14.	private GoogleMap googleMap;
15.	
16.	// Nilai disimpan yang didapatkan dari FilterActivity
17.	private String data, type, parameter, parameterValue;
18.	
19.	private ProgressDialog progressDialog;
20.	
21.	// Map untuk menyimpan detail dari suatu Polygon
22.	private Map<String, Pair<String, String>> polygonDataRef;
23.	
24.	// List untuk menyimpan seluruh listOf Marker yang ada di Maps
25.	private List<Marker> markerSet;
26.	
27.	@Override
28.	protected void onCreate(Bundle savedInstanceState) {
29.	super.onCreate(savedInstanceState);
30.	
31.	setContentView(R.layout.activity_maps);
32.	
33.	filterButton = (FloatingActionButton)
34.	findViewById(R.id.filterButton);
35.	helperIndicator = (FrameLayout)
36.	findViewById(R.id.helperIndicator);
37.	
38.	DrawerLayout drawer = (DrawerLayout)
39.	findViewById(R.id.drawer_layout);
40.	ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
41.	this, drawer, null, R.string.navigation_drawer_open,
42.	R.string.navigation_drawer_close);
43.	drawer.setDrawerListener(toggle);
44.	toggle.syncState();
45.	
46.	NavigationView navigationView = (NavigationView)
47.	findViewById(R.id.nav_view);
48.	navigationView.setNavigationItemSelectedListener(this);
49.	
50.	SupportMapFragment mapFragment = (SupportMapFragment)
51.	getSupportFragmentManager()
52.	.findFragmentById(R.id.map);
53.	
54.	// Register callback saat map ready
55.	mapFragment.getMapAsync(this);

```
56.
57.         // Set filterButton disabled, akan di enable saat maps ready
58.         filterButton.setEnabled(false);
59.
60.         filterButton.setOnClickListener(new View.OnClickListener() {
61.             @Override
62.             public void onClick(View v) {
63.                 showFilterDialog();
64.             }
65.         });
66.     }
67.
68.     @Override
69.     public void onMapReady(GoogleMap googleMap) {
70.         this.googleMap = googleMap;
71.
72.         // Set camera dari MapsView agar mengarah ke daerah Malang
73.         LatLng startingPoint = new LatLng(-7.988137557998129,
74. 112.64743995681773);
75.
76. this.googleMap.moveCamera(CameraUpdateFactory.newLatLngZoom(startingPo
77. int, 12.0f));
78.
79.         // Nonaktifkan tombol yang disediakan oleh GoogleMaps
80.         this.googleMap.getUiSettings().setMapToolbarEnabled(false);
81.
82.         // Menunggu hingga layout helperIndicator selesai di tampilkan
83.
84. helperIndicator.getViewTreeObserver().addOnGlobalLayoutListener(new
85. ViewTreeObserver.OnGlobalLayoutListener() {
86.             @Override
87.             public void onGlobalLayout() {
88.                 RelativeLayout.LayoutParams params =
89. (RelativeLayout.LayoutParams) helperIndicator.getLayoutParams();
90.
91.                 // Set padding pada googleMap agar logo Google tidak
92. tertutup oleh helperIndicator
93.                 Maps.this.googleMap.setPadding(0, 0, 0,
94. helperIndicator.getHeight() + params.bottomMargin);
95.             }
96.         });
97.
98.         // Aktifkan filterButton
99.         filterButton.setEnabled(true);
100.
101.         // Daftarkan listener saat polygon di maps di tekan
102.         googleMap.setOnPolygonClickListener(new
103. GoogleMap.OnPolygonClickListener() {
104.             @Override
105.             public void onPolygonClick(Polygon polygon) {
106.                 // Apabila markerSet null, buat object ArrayList
107.                 if (markerSet == null) {
108.                     markerSet = new ArrayList<>();
109.                 }
110.
111.                 // Hapus seluruh marker yang ada pada markerSet
112.                 for (Marker marker : markerSet) {
113.                     marker.remove();
114.                 }
```

```
115.                 markerSet.clear();
116.
117.                 // Kalau polygonDataRef bukan null
118.                 if (polygonDataRef != null) {
119.                     // Buat LatLngBounds untuk kalkulasi titik tengah
120. dari suatu region
121.                     LatLngBounds.Builder boundsBuilder = new
122. LatLngBounds.Builder();
123.                     for (LatLng point : polygon.getPoints()) {
124.                         boundsBuilder.include(point);
125.                     }
126.                     LatLngBounds bounds = boundsBuilder.build();
127.
128.                     // Ambil data region pada polygonDataRef sesuai id
129. polygon
130.                     Pair<String, String> data =
131. polygonDataRef.get(polygon.getId());
132.
133.                     // Kalau data tidak null
134.                     if (data != null) {
135.                         // Buat marker dengan judul data, tanpa icon
136.                         Marker marker = Maps.this.googleMap.addMarker(
137.                                 new MarkerOptions()
138.                                         .position(bounds.getCenter())
139.                                         .title(data.first)
140.                                         .snippet(data.second)
141.
142. .icon(BitmapDescriptorFactory.fromResource(R.drawable.ic_blank))
143.                         );
144.                         // Tampilkan window info
145.                         marker.showInfoWindow();
146.
147.                         // tambahkan marker ke dalam markerSet
148.                         markerSet.add(marker);
149.                     }
150.                 }
151.             }
152.         });
153.     }
154.
155.     private void showFilterDialog() {
156.         // Buat Intent yang bertujuan untuk membuka FilterActivity
157.         Intent intent = new Intent(this, Filter.class);
158.
159.         // Tambahkan data yang ada pada variable diatas seusai dengan
160. Tag-nya
161.         intent.putExtra(Filter.DATA_TAG, data);
162.         intent.putExtra(Filter.TYPE_TAG, type);
163.         intent.putExtra(Filter.PARAMETER_TAG, parameter);
164.         intent.putExtra(Filter.PARAMETER_VALUE_TAG, parameterValue);
165.
166.         // Buka Activity FilterActivity
167.         startActivityForResult(intent, FILTER_REQUEST_KEY);
168.     }
169.
170.     @Override
171.     protected void onActivityResult(int requestCode, int resultCode,
172. Intent intent) {
173.         super.onActivityResult(requestCode, resultCode, intent);
```

```
174.
175.         // Apabila ada response dari FilterActivity dengan result OK
176.         if (requestCode == FILTER_REQUEST_KEY && resultCode ==
177. RESULT_OK) {
178.             data = intent.getStringExtra(Filter.DATA_TAG);
179.             type = intent.getStringExtra(Filter.TYPE_TAG);
180.             parameter = intent.getStringExtra(Filter.PARAMETER_TAG);
181.             parameterValue =
182. intent.getStringExtra(Filter.PARAMETER_VALUE_TAG);
183.
184.             // Set data variable
185.             if (data == null) data = "";
186.             if (type == null) type = "";
187.             if (parameter == null) parameter = "";
188.             if (parameterValue == null) parameterValue = "";
189.
190.             // Update Maps
191.             handleUpdateMaps();
192.         }
193.     }
194.
195.     private void handleUpdateMaps() {
196.         // Hapus segala Polygon dan Marker yang ada di maps
197.         googleMap.clear();
198.
199.         // Apabila ada progress Dialog yang sedang tampil, maka tutup
200. progress Dialog tersebut
201.         if (progressDialog != null && progressDialog.isShowing()) {
202.             progressDialog.dismiss();
203.         }
204.
205.         // Buat progress Dialog
206.         progressDialog = ProgressDialog.show(this, "", "Mohon
207. tunggu...", true, false);
208.
209.         // Apabila data adalah datatb
210.         if (data.equalsIgnoreCase("datatb")) {
211.             // Panggil fungsi untuk getDataTbCalculation, kemudian
212. konversikan menggunakan fungsi mapper
213.             // Response apabila sukses, eksekusi consumeListOfRegion,
214. apabila ada error, eksekusi consumeThrowable
215.             FunctionsProvider.getInstance().getDataTbCalculation(type,
216. parameter, parameterValue)
217.                     .map(mapper)
218.                     .subscribeOn(Schedulers.io())
219.                     .observeOn(AndroidSchedulers.mainThread())
220.                     .subscribe(consumeListOfRegion, consumeThrowable);
221.         } else if (data.equalsIgnoreCase("datagiziburuk")) {
222.             // Panggil fungsi untuk getDataGiziBurukCalculation,
223. kemudian konversikan menggunakan fungsi mapper
224.             // Response apabila sukses, eksekusi consumeListOfRegion,
225. apabila ada error, eksekusi consumeThrowable
226.
227. FunctionsProvider.getInstance().getDataGiziBurukCalculation(type,
228. parameter, parameterValue)
229.                     .map(mapper)
230.                     .subscribeOn(Schedulers.io())
231.                     .observeOn(AndroidSchedulers.mainThread())
232.                     .subscribe(consumeListOfRegion, consumeThrowable);
```

```
233.         } else if (data.equalsIgnoreCase("databumil")) {
234.             // Panggil fungsi untuk getDataBumilCalculation, kemudian
235. konversikan menggunakan fungsi mapper
236.             // Response apabila sukses, eksekusi consumeListOfRegion,
237. apabila ada error, eksekusi consumeThrowable
238. 
239. FunctionsProvider.getInstance().getDataBumilCalculation(type,
240. parameter, parameterValue)
241.                     .map(mapper)
242.                     .subscribeOn(Schedulers.io())
243.                     .observeOn(AndroidSchedulers.mainThread())
244.                     .subscribe(consumeListOfRegion, consumeThrowable);
245.         } else {
246.             // Apabila tidak terpenuhi, hapus progress Dialog
247.             if (progressDialog != null && progressDialog.isShowing())
248. {
249.                 progressDialog.dismiss();
250.             }
251.         }
252.     }
253. 
254.     private Function<CalculationResponse, List<Pair<Region, Integer>>>
255. mapper = new Function<CalculationResponse, List<Pair<Region,
256. Integer>>>() {
257.         @Override
258.         public List<Pair<Region, Integer>> apply(CalculationResponse
259. calculationResponse) throws Exception {
260.             List<Region> regionList;
261.             // Cek type, kalau kecamatan, ambil seluruh kecamatan
262. region, kalau kelurahan, ambil seluruh kelurahan region
263.             if (type.equalsIgnoreCase("kecamatan")) {
264.                 regionList =
265. RegionProvider.getInstance(Maps.this).getAllKecamatanRegions();
266.             } else {
267.                 regionList =
268. RegionProvider.getInstance(Maps.this).getAllKelurahanRegions();
269.             }
270. 
271.             // Buat listOf Region
272.             List<Pair<Region, Integer>> newRegion = new ArrayList<>();
273.             for (Region region : regionList) {
274.                 int total = 0;
275.                 double percentage = 0.0;
276.                 double density = 0.0;
277.                 int color;
278.                 String label;
279. 
280.                 // Cari DetailData yang sesuai dengan Region saat ini
281.                 for (CalculationResponse.DetailData detailData :
282. calculationResponse.getDetail()) {
283.                     if
284. (detailData.getName().equalsIgnoreCase(region.getName())) {
285.                         // Ambil nilai total, percentage, dan density
286.                         total = detailData.getTotal();
287.                         percentage = detailData.getPercentage();
288.                         density = detailData.getDensity();
289.                         break;
290.                     }
291.                 }
```

```
292.
293.                 // Penilaian density
294.                 if (density < 0.2) {
295.                     color = ContextCompat.getColor(Maps.this,
296. R.color.rendahSekali_color);
297.                     label =
298. getResources().getString(R.string.rendahSekali_label);
299.                 } else if (density < 0.4) {
300.                     color = ContextCompat.getColor(Maps.this,
301. R.color.rendah_color);
302.                     label =
303. getResources().getString(R.string.rendah_label);
304.                 } else if (density < 0.6) {
305.                     color = ContextCompat.getColor(Maps.this,
306. R.color.menengah_color);
307.                     label =
308. getResources().getString(R.string.menengah_label);
309.                 } else if (density < 0.8) {
310.                     color = ContextCompat.getColor(Maps.this,
311. R.color.tinggi_color);
312.                     label =
313. getResources().getString(R.string.tinggi_label);
314.                 } else {
315.                     color = ContextCompat.getColor(Maps.this,
316. R.color.tinggiSekali_color);
317.                     label =
318. getResources().getString(R.string.tinggiSekali_label);
319.                 }
320.
321.                 if (density == 0.0) {
322.                     color = ContextCompat.getColor(Maps.this,
323. R.color.kosong_color);
324.                     label =
325. getResources().getString(R.string.kosong_label);
326.                 }
327.
328.                 // Tambahkan region tersebut dan total data ke
329. newRegion
330.                 Region parsed = new Region(region.getName(), total,
331. percentage, density, color, label, region.getAreaList());
332.
333.                 newRegion.add(new Pair<Region, Integer>(parsed,
334. calculationResponse.getTotalData()));
335.             }
336.
337.             // Kembalikan region
338.             return newRegion;
339.         }
340.     };
341.
342.     private Consumer<List<Pair<Region, Integer>>> consumeListOfRegion
343. = new Consumer<List<Pair<Region, Integer>>>() {
344.         @Override
345.         public void accept(List<Pair<Region, Integer>> regions) throws
346. Exception {
347.             // Hapus seluruh object yang ada di maps
348.             googleMap.clear();
349.
350.             // Kalau polygonDataRef null, buat HashMap
```

```
            if (polygonDataRef == null) {
                polygonDataRef = new HashMap<>();
            }

            // Hapus isi polygonDataRef
            polygonDataRef.clear();

            // Looping dari regions yang ada
            for (int i = 0; i < regions.size(); i++) {
                // Ambil listOf LatLng yang ada pada region tersebut
                List<LatLng> latLngList =
regions.get(i).first.getAreaList();

                // Buat polygonnya
                Polygon addedPolygon = googleMap.addPolygon(new
PolygonOptions()
                        .add(latLngList.toArray(new
LatLng[latLngList.size()]))
                        .strokeColor(ContextCompat.getColor(Maps.this,
R.color.border_color))
                        .strokeWidth(2.5f)
                        .clickable(true)
                        .fillColor(regions.get(i).first.getColor()));

                // Tambahkan data polygon ke polygonDataRef
                String statistics = "Terdapat " +
regions.get(i).first.getTotal() + " data dari " +
regions.get(i).second + " data yang ada.";

                polygonDataRef.put(addedPolygon.getId(), new
Pair<>(regions.get(i).first.getName(), statistics));
            }

            // Kalau progress Dialog sedang tampil, tutup progess
Dialog tersebut
            if (progressDialog != null && progressDialog.isShowing())
{
                progressDialog.dismiss();
            }
        }
    };

    private Consumer<Throwable> consumeThrowable = new
Consumer<Throwable>() {
        @Override
        public void accept(Throwable throwable) throws Exception {
            // Tampilkan pesan error
            Toast.makeText(Maps.this, "Error: " +
throwable.getLocalizedMessage(), Toast.LENGTH_SHORT).show();

            // Kalau progress Dialog sedang tampil, tutup progess
Dialog tersebut
            if (progressDialog != null && progressDialog.isShowing())
{
                progressDialog.dismiss();
            }
        }
    };

```

```
410.     @Override
411.     public void onBackPressed() {
412.         DrawerLayout drawer = (DrawerLayout)
413. findViewById(R.id.drawer_layout);
414.         if (drawer.isDrawerOpen(GravityCompat.START)) {
415.             drawer.closeDrawer(GravityCompat.START);
416.         } else {
417.             super.onBackPressed();
418.         }
419.     }
420.
421.     @Override
422.     public boolean onCreateOptionsMenu(Menu menu) {
423.         // Inflate the menu; this adds items to the action bar if it
424. is present.
425.         getMenuInflater().inflate(R.menu.home, menu);
426.         return true;
427.     }
428.
429.     @Override
430.     public boolean onOptionsItemSelected(MenuItem item) {
431.         // Handle action bar item clicks here. The action bar will
432.         // automatically handle clicks on the Home/Up button, so long
433.         // as you specify a parent activity in AndroidManifest.xml.
434.         int id = item.getItemId();
435.
436.         //noinspection SimplifiableIfStatement
437.         if (id == R.id.action_settings) {
438.             return true;
439.         }
440.
441.         return super.onOptionsItemSelected(item);
442.     }
443.
444.     @SuppressWarnings("StatementWithEmptyBody")
445.     @Override
446.     public boolean onNavigationItemSelected(MenuItem item) {...}
447.     public boolean onKeyDown(int keyCode, KeyEvent event) {...}
448.     protected void exitByBackKey() {...}
449. }
450.
451.
```

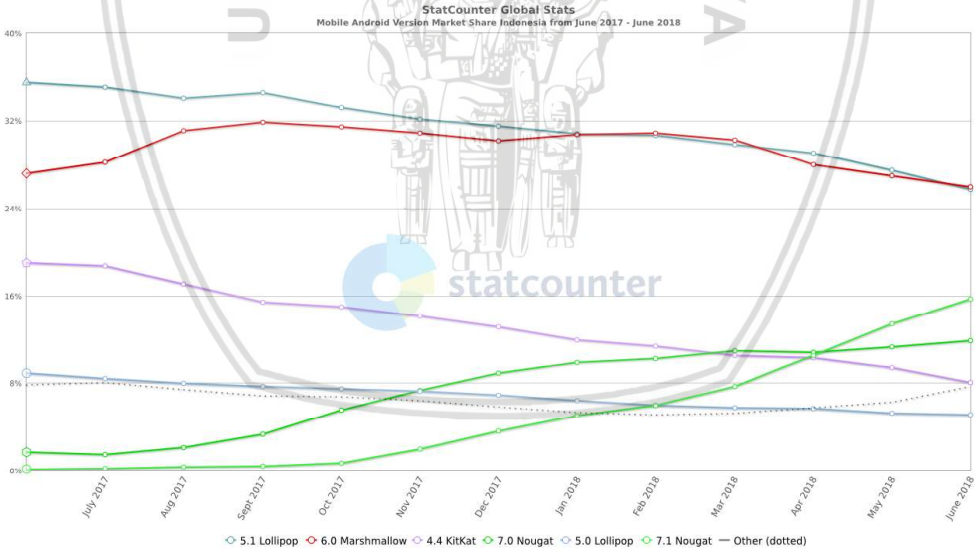
BAB 7 PENGUJIAN DAN ANALISIS

Bab ini membahas tentang pengujian setelah tahap implementasi dilakukan dari aplikasi informasi kesehatan masyarakat kota malang berbasis *mobile native android*. Pengujian ini bertujuan untuk mengetahui bahwa aplikasi telah memenuhi kebutuhan pengguna yang didefinisikan di awal. Pengujian yang dilakukan meliputi Pengujian Unit, Validasi, dan *Compatibility*.

7.1 Spesifikasi Perangkat Uji

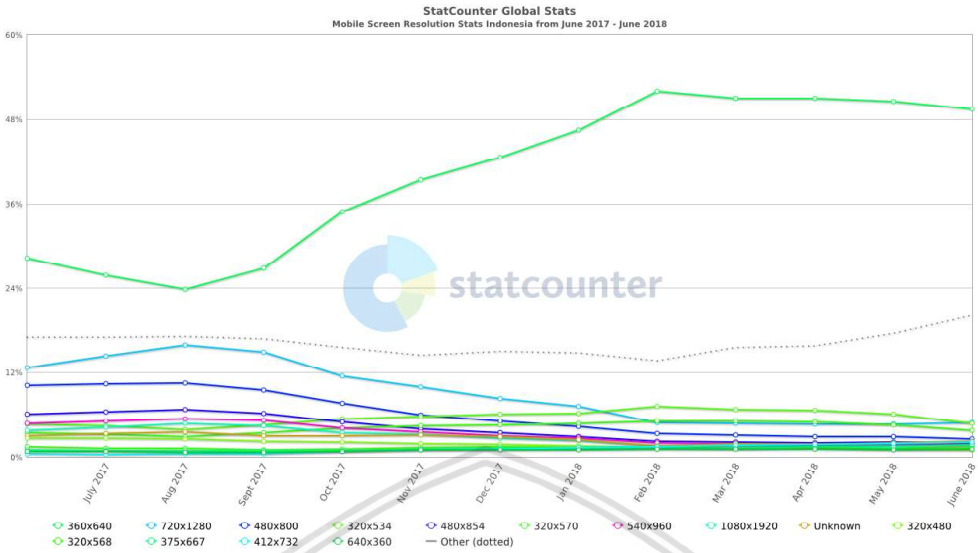
Pengujian ini menggunakan dua jenis perangkat keras yang telah disiapkan dan empat jenis perangkat virtual yang disiapkan dengan perbedaan spesifikasi. Perangkat tersebut mewakili dua populasi terbanyak untuk dijadikan sampel dengan parameter versi sistem operasi *android* yang berbeda, dan ukuran jenis layar yang berbeda. Data populasi tersebut diperoleh dari *website StatCounter* (StatCounter, 2017).

Menurut (StatCounter, 2017), dua versi sistem operasi *android* yang banyak digunakan di Indonesia adalah android versi *6.0 Marshmallow* sebesar 25,92%, kemudian versi 5.1 *Lollipop* sebesar 25,71% per juni 2017 hingga juni 2018. Seperti pada Gambar 7.1 yang dapat disimpulkan bahwa populasi terbanyak pada versi *android Marshmallow* dan *lollipop*.



Gambar 7.1 Populasi *Android Operating System Version*

Pada dua jenis populasi terbanyak dari parameter ukuran layar menurut (StatCounter, 2017), di Indonesia sebanyak 49,4% menggunakan resolusi *360x640 px*, dan *720x1280 px* digunakan sebanyak 4,82%. Seperti yang ditunjukkan pada Gambar 7.2 dibawah.



Gambar 7.2 Populasi *Screen Resolution*

Perangkat uji yang digunakan merupakan perangkat uji yang mewakili dari semua jenis versi sistem operasi android serta ukuran layar yang digunakan oleh masyarakat Indonesia berdasarkan perolehan data pada Gambar 7.1 dan Gambar 7.2. Penjelasan spesifikasi perangkat uji yang digunakan dalam pengujian aplikasi informasi kesehatan masyarakat kota Malang berbasis *mobile native android* ini akan dijelaskan pada Tabel 7.1 sampai Tabel 7.6 sebagai berikut:

Tabel 7.1 Spesifikasi Perangkat Keras ASUS Zenfone Pegasus 3

No.	Jenis Perangkat Uji	Spesifikasi
1.	Model	ASUS Zenfone Pegasus 3 (ASUS_X008DA)
2.	Chipset	Mediatek MT6737
3.	Sistem Operasi	Android Nougat 7.0
4.	Memory (RAM)	2 GB
5.	Memory Storage	16 GB
6.	Processor (CPU)	Quad-core 1.3 GHz Cortex-A53
7.	Grapic (GPU)	Mali-T720MP2
8.	Resolution	720 x 1280 pixels, 16:9 ratio
9.	Density resolution	282 ppi density

Tabel 7.2 Spesifikasi Perangkat Keras Xiaomi Redmi 3s Prime

No.	Jenis Perangkat Uji	Spesifikasi
1.	Model	Huawei Mate 10 Lite
2.	Chipset	HiSilicon Kirin 659
3.	Sistem Operasi	Android Nougat 7.0
4.	Memory (RAM)	4 GB
5.	Memory Storage	64 GB
6.	Processor (CPU)	Octa-core (4x2.36 GHz Cortex-A53 & 4x1.7 GHz Cortex-A53)
7.	Grapic (GPU)	Mali-T830 MP2
8.	Resolution	1080 x 2160 pixels, 18:9 ratio
9.	Density resolution	409 ppi density

Tabel 7.3 Spesifikasi *Android Virtual Device 1*

No.	Jenis Perangkat Uji	Spesifikasi
1.	Model	Android Virtual Device
2.	Chipset	-
3.	Sistem Operasi	Android Lollipop 5.1
4.	Memory (RAM)	2 GB
5.	Memory Storage	-
6.	Processor (CPU)	-
7.	Grapic (GPU)	-
8.	Resolution	360x640 pixels
9.	Density resolution	-

Tabel 7.4 Spesifikasi *Android Virtual Device 2*

No.	Jenis Perangkat Uji	Spesifikasi
1.	Model	Android Virtual Device
2.	Chipset	-
3.	Sistem Operasi	Android Lollipop 5.1
4.	Memory (RAM)	2 GB
5.	Memory Storage	-
6.	Processor (CPU)	-
7.	Grapic (GPU)	-
8.	Resolution	720x1280 pixels
9.	Density resolution	-

Tabel 7.5 Spesifikasi *Android Virtual Device 3*

No.	Jenis Perangkat Uji	Spesifikasi
1.	Model	Android Virtual Device
2.	Chipset	-
3.	Sistem Operasi	Android Marshmallow 6.0
4.	Memory (RAM)	2 GB
5.	Memory Storage	-
6.	Processor (CPU)	-
7.	Grapic (GPU)	-
8.	Resolution	360x640 pixels
9.	Density resolution	-

Tabel 7.6 Spesifikasi *Android Virtual Device 4*

No.	Jenis Perangkat Uji	Spesifikasi
1.	Model	Android Virtual Device
2.	Chipset	-
3.	Sistem Operasi	Android Marshmallow 6.0
4.	Memory (RAM)	2 GB
5.	Memory Storage	-
6.	Processor (CPU)	-
7.	Grapic (GPU)	-
8.	Resolution	720x1280pixels
9.	Density resolution	-

7.2 Pengujian Unit

Pengujian unit dilakukan dengan metode *Whitebox testing* untuk mengetahui kompleksitas fungsi serta algoritma sistem yang telah implementasikan. *Whitebox testing* disini menggunakan *basis path* untuk menghitung *cyclomatic complexity* dari setiap fitur yang disediakan oleh aplikasi dengan menghitung rumus *V (G) = E – N + 2* (Pressman, 2010). Perhitungan rumus ini akan diaplikasikan pada algoritma fitur *add, edit, delete, view,* dan *maps.*

7.2.1 Pengujian Unit Algoritma

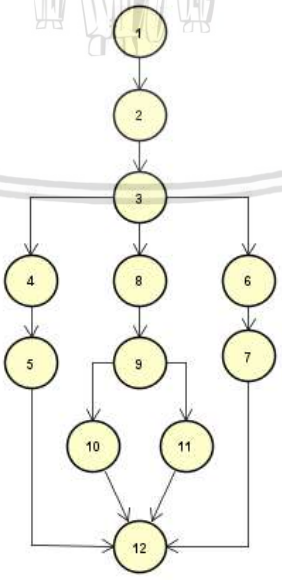
Pengujian Unit Algoritma akan dijelaskan pada Sub bab dari bab 7.2.1 sebagai berikut.

7.2.1.1 Pengujian Unit Algoritma *Add* Data

Pengujian unit algoritma *add* data akan dijelaskan pada Tabel 7.7.

Tabel 7.7 Tabel Pengujian Unit Algoritma *Add* Data

Algoritma *Add* Data
Mulai___(1) User menekan tombol Management Data___(2) Tampil list data___(3) If user menekan list data___(4) Tampil data___(5) Else user menekan tombol search___(6) Tampil form search___(7) Else user menekan tombol add data___(8) Tampil form add data___(9) If user menekan tombol batal___(10) Else user menekan tombol simpan___(11) Selesai___(12)



Gambar 7.3 *Flow Graph* Algoritma *Add* Data

Pada Gambar 7.3 menunjukkan *Flow Graph* algoritma *add* data yang dapat dihitung *cyclomatic complexity* nya sebagai berikut:

V (G) = E – N + 2
= 14 – 12 + 2
= 4

Dari hasil perhitungan diatas ditemukan hasil sebagai nilai jumlah *independent path* yang tersedia yaitu :

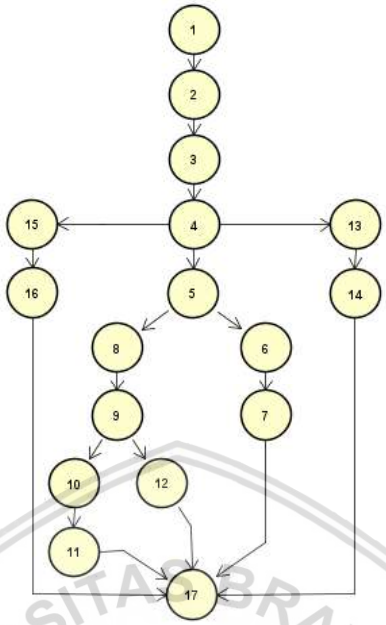
Path 1 : 1-2-3-8-9-10-12
Path 2 : 1-2-3-8-9-11-12
Path 3 : 1-2-3-4-5-12
Path 4 : 1-2-3-6-7-12

7.2.1.2 Pengujian Unit Algoritma *Edit* Data

Pengujian unit algoritma *edit* data akan dijelaskan pada Tabel 7.8.

Tabel 7.8 Tabel Pengujian Unit Algoritma *Edit* Data

Algoritma *Edit* Data
Mulai___(1) User menekan tombol Management Data___(2) Tampil list data___(3) If user menekan list data___(4) Tampil data___(5) If user menekan tombol delete___(6) Delete data___(7) Else user menekan tombol edit data___(8) Tampil form Edit data___(9) If user menekan tombol batal___(10) Batal___(11) Else user menekan tombol simpan___(12) Else user menekan tombol search___(13) Tampil form search___(14) Else user menekan tombol add data___(15) Tampil form add data___(16) Selesai___(17)



Gambar 7.4 *Flow Graph* Algoritma *Edit* Data

Pada Gambar 7.4 menunjukkan *Flow Graph* algoritma *edit* data yang dapat dihitung *cyclomatic complexity* nya sebagai berikut:

V (G) = E – N + 2
= 20 – 17 + 2
= 5

Dari hasil perhitungan diatas ditemukan hasil sebagai nilai jumlah *independent path* yang tersedia yaitu :

Path 1 : 1-2-3-4-15-16-17
Path 2 : 1-2-3-4-5-8-9-10-11-17
Path 3 : 1-2-3-4-5-8-9-12-17
Path 4 : 1-2-3-4-5-6-7-17
Path 5 : 1-2-3-4-13-14-17

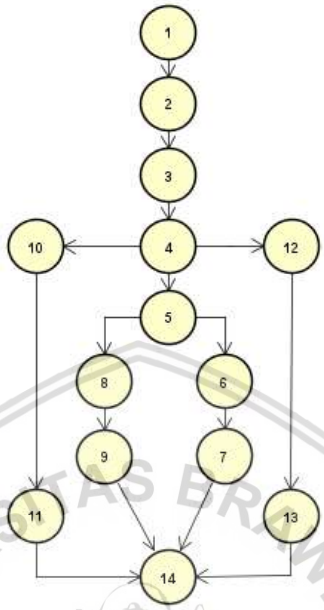
7.2.1.3 Pengujian Unit Algoritma *Delete* Data

Pengujian unit algoritma *delete* data akan dijelaskan pada Tabel 7.9.

Tabel 7.9 Tabel Pengujian Unit Algoritma *Delete* Data

Algoritma *Delete* Data
Mulai___(1) User menekan tombol Management Data___(2) Tampil list data___(3) If user menekan list data___(4) Tampil data___(5) If user menekan tombol delete___(6) Delete data___(7) Else user menekan tombol edit data___(8) Tampil form Edit data___(9) Else user menekan tombol search___(10) Tampil form search___(11)

```
          Else user menekan tombol add data___(12)
               Tampil form add data___(13)
Selesai   (14)
```



Gambar 7.5 *Flow Graph* Algoritma *Delete* Data

Pada Gambar 7.5 menunjukkan *Flow Graph* algoritma *delete* data yang dapat dihitung *cyclomatic complexity* nya sebagai berikut:

V (G) = E – N + 2
= 16 – 14 + 2
= 4

Dari hasil perhitungan diatas ditemukan hasil sebagai nilai jumlah *independent path* yang tersedia yaitu :

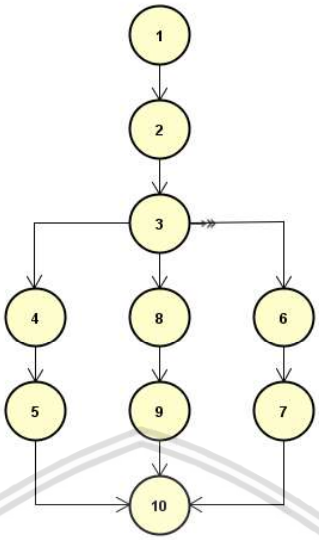
Path 1 : 1-2-3-4-10-11-14
Path 2 : 1-2-3-4-5-8-9-14
Path 3 : 1-2-3-4-5-6-7-14
Path 4 : 1-2-3-4-12-13-14

7.2.1.4 Pengujian Unit Algoritma *View* Data

Pengujian unit algoritma *view* data akan dijelaskan pada Tabel 7.10.

Tabel 7.10 Tabel Pengujian Unit Algoritma *View* Data

Algoritma *View* Data
Mulai___(1) User menekan tombol Management Data___(2) Tampil list data___(3) If user menekan list data___(4) Tampil data___(5) Else user menekan tombol search___(6) Tampil form search___(7) Else user menekan tombol add data___(8) Tampil form add data___(9) Selesai___(10)



Gambar 7.6 *Flow Graph* Algoritma *View* Data

Pada Gambar 7.6 menunjukkan *Flow Graph* algoritma *view* data yang dapat dihitung *cyclomatic complexity* nya sebagai berikut:

V (G) = E – N + 2
= 11 – 10 + 2
= 3

Dari hasil perhitungan diatas ditemukan hasil sebagai nilai jumlah *independent path* yang tersedia yaitu :

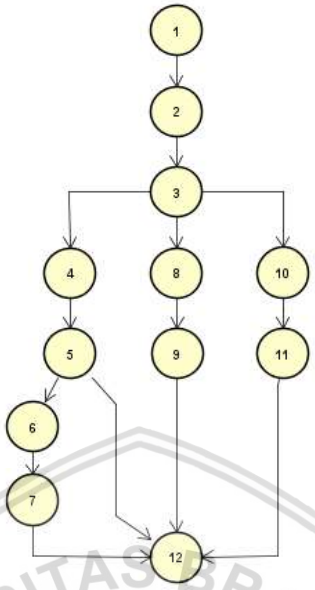
Path 1 : 1-2-3-4-5-10
Path 2 : 1-2-3-8-9-10
Path 3 : 1-2-3-6-7-10

7.2.1.5 Pengujian Unit Algoritma *Maps*

Pengujian unit algoritma *maps* data akan dijelaskan pada Tabel 7.11.

Tabel 7.11 Tabel Pengujian Unit Algoritma *Maps* Data

Algoritma *Maps* Data
Mulai___(1) User menekan tombol navigationpanel___(2) Tampil menu navigation___(3) If user menekan maps___(4) Tampil maps___(5) If user menekan tombol filter___(6) Tampil filter maps___(7) Else user menekan tombol management data___(8) Tampil management data___(9) Else user menekan tombol logout___(10) Logout___(11) Selesai___(12)



Gambar 7.7 *Flow Graph* Algoritma *Maps* Data

Pada Gambar 7.7 menunjukkan *Flow Graph* algoritma *maps* data yang dapat dihitung *cyclomatic complexity* nya sebagai berikut:

V (G) = E – N + 2
= 14 – 12 + 2
= 4

Dari hasil perhitungan diatas ditemukan hasil sebagai nilai jumlah *independent path* yang tersedia yaitu :

Path 1 : 1-2-3-4-5-6-7-12
Path 2 : 1-2-3-5-12
Path 3 : 1-2-3-8-12
Path 4 : 1-2-3-10-11-12

7.2.2 Hasil Analisis Pengujian Unit

Hasil pengujian unit dengan menggunakan metode *Whitebox* yang telah dilakukan dengan menghitung *cyclomatic complexity* telah sesuai dengan *independent path* yang terdapat pada algoritma fitur aplikasi. Selain itu hasil pengujian ini sudah sesuai dengan perancangan dan implementasi aplikasi yang sebelumnya telah di lakukan. Dapat disimpulkan bahwa aplikasi dapat berjalan dengan baik dan sesuai seperti yang ditunjukkan pada Tabel 7.12.

Tabel 7.12 Tabel Hasil Pengujian Unit

	Add	***Edit***	***Delete***	***View***	***Maps***
Cyclomatic Complexity	4	5	4	3	4
Total Basis Path	4	5	4	3	4
Status	Valid	Valid	Valid	Valid	Valid

7.3 Pengujian Fungsional

Pengujian fungsional merupakan pengujian yang dilakukan untuk memastikan kebutuhan-kebutuhan pokok pada aplikasi yang didefinisikan diawal telah sesuai dengan kebutuhan aplikasi untuk selanjutnya divalidasi. Pengujian ini menggunakan metode pengujian *Blackbox testing* dengan acuan yang telah ditentukan skenario *usecase* sebelumnya (Pressman, 2010).

Pengujian fungsional akan dilakukan dengan cara mendeskripsi kasus uji dan akan dilakukan proses pengujian yang kemudian akan dibandingkan dengan hasil pengujian. Selanjutnya akan dihitung prosentase yang merupakan analisa kelayakan aplikasi.

7.3.1 Kasus Uji Pengujian Fungsional

Pada sub bab 7.3.1 ini kasus uji akan didefiniskan terlebih dahulu dan dijelaskan pada Tabel 7.13 hingga Tabel 7.19.

Tabel 7.13 Kasus Uji *Login*

Nomor Kasus Uji	SIMKES-Fung-01
Nama Kasus Uji	Kasus Uji *Login*
Tujuan Pengujian	Pengujian dilakukan untuk memastikan aplikasi dapat menampilkan halaman awal / halaman utama pada aplikasi setelah melakukan *login* dengan memasukkan *email* dan *password* dengan benar
Kasus Uji	Pengujian dilakukan dengan memasukkan kombinasi *email* dan *password* yang sesuai
Prosedur pengujian	1. Penguji membuka aplikasi *SIMKES* dan masuk kehalaman *login* 2. Penguji memasukkan kombinasi *email* dan *password* yang benar.
Hasil yang diharapkan	Aplikasi menampilkan halaman utama aplikasi setelah berhasil melakukan *login*.

Tabel 7.14 Kasus Uji *View* Data

Nomor Kasus Uji	SIMKES-Fung-02
Nama Kasus Uji	Kasus Uji *View* Data
Tujuan Pengujian	Pengujian dilakukan untuk memastikan aplikasi dapat menampilkan data setelah menekan menu management data dan memilih salah satu data
Kasus Uji	Pengujian dilakukan dengan menekan salah satu data dari list data yang ada
Prosedur pengujian	1. Penguji menekan salah satu data yang tampil pada list data di halaman management data
Hasil yang diharapkan	Aplikasi menampilkan informasi mendetail tentang data yang dipilih sebelumnya

Tabel 7.15 Kasus Uji *Delete* Data

Nomor Kasus Uji	SIMKES-Fung-03
Nama Kasus Uji	Kasus Uji *Delete* Data
Tujuan Pengujian	Pengujian dilakukan untuk memastikan aplikasi dapat menghapus data setelah menekan tombol “hapus”

Kasus Uji	Pengujian dilakukan dengan menekan tombol “hapus” setelah melihat data yang telahdipilih sebelumnya
Prosedur pengujian	1. Penguji menekan salah satu data yang tampil pada list data di halaman management data 2. Penguji menekan tombol “hapus” untuk menghapus salah satu data
Hasil yang diharapkan	Aplikasi telah menghapus data tersebut

Tabel 7.16 Kasus Uji *Add* Data

Nomor Kasus Uji	SIMKES-Fung-04
Nama Kasus Uji	Kasus Uji *Add* Data
Tujuan Pengujian	Pengujian dilakukan untuk memastikan aplikasi dapat menyimpan data setelah menekan tombol “Simpan”
Kasus Uji	Pengujian dilakukan dengan menekan tombol “Simpan” setelah mengisi atribut data yang ada
Prosedur pengujian	1. Penguji menekan tombol *add* data 2. Penguji mengisi form atribut data yang dibutuhkan 3. Penguji menekan tombol “Simpan”
Hasil yang diharapkan	Aplikasi telah menyimpan data tersebut

Tabel 7.17 Kasus Uji *Edit* Data

Nomor Kasus Uji	SIMKES-Fung-05
Nama Kasus Uji	Kasus Uji *Edit* Data
Tujuan Pengujian	Pengujian dilakukan untuk memastikan aplikasi dapat menyimpan data setelah mengubah data dan menekan tombol “Simpan”
Kasus Uji	Pengujian dilakukan dengan menekan tombol “Simpan” setelah mengubah atribut data yang ada
Prosedur pengujian	1. Penguji menekan tombol *edit* data pada halaman *view* data 2. Penguji mengubah form atribut data yang dibutuhkan 3. Penguji menekan tombol “Simpan”
Hasil yang diharapkan	Aplikasi telah menyimpan data yang telah dirubah tersebut

Tabel 7.18 Kasus Uji *Maps*

Nomor Kasus Uji	SIMKES-Fung-06
Nama Kasus Uji	Kasus Uji *Maps*
Tujuan Pengujian	Pengujian dilakukan untuk memastikan aplikasi dapat menampilkan halaman *maps* pada aplikasi setelah melakukan filter data *maps* yang sudah disediakan dengan benar
Kasus Uji	Pengujian dilakukan dengan membuka halaman *maps* dan melakukan filter data *maps*
Prosedur pengujian	1. Penguji membuka menu maps 2. Penguji menekan tombol filter data *maps* 3. Penguji memasukkan filter data *maps* yang telah disediakan 4. Penguji menekan tombol *checklist* untuk menampilkan filter data *maps* yang telah dilakukan
Hasil yang diharapkan	Aplikasi menampilkan *maps* dengan perbedaan warna setiap tingkat persebaran data yang telah difilter sebelumnya

Tabel 7.19 Kasus Uji *Logout*

Nomor Kasus Uji	SIMKES-Fung-07
Nama Kasus Uji	Kasus Uji *Logout*
Tujuan Pengujian	Pengujian dilakukan untuk memastikan aplikasi dapat melakukan proses *logout* setelah menekan tombol logout oada navigation panel
Kasus Uji	Pengujian dilakukan dengan menekan tombol *logout*
Prosedur pengujian	1. Penguji menekan tombol logout untuk keluar dari aplikasi
Hasil yang diharapkan	Aplikasi menampilkan halaman *login* setelah berhasil melakukan *logout.*

7.3.2 Hasil Analisis Pengujian Fungsional

Hasil pengujian fungsional akan dijelaskan menggunakan tabel hasil pengujian fungsional yang nantinya akan menjadi kesimpulan untuk pengujian fungsional. Berikut merupakan penjelasan secara mendetail dari hasil pengujian fungsional akan ditunjukkan pada Tabel 7.20.

Tabel 7.20 Tabel Hasil Pengujian Fungsional

No. Kasus Uji	Nama Kasus Uji	Hasil yang diharapkan	Hasil Pengujian	Status Perangkat					
				1	2	3	4	5	6
SIMKES-Fung-01	Kasus Uji *Login*	Aplikasi menampilkan halaman utama aplikasi setelah berhasil melakukan *login.*	Aplikasi menampilkan halaman utama aplikasi setelah berhasil melakukan *login.*	v	v	v	v	v	v
SIMKES-Fung-02	Kasus Uji *View* Data	Aplikasi menampilkan informasi mendetail tentang data yang dipilih sebelumnya	Aplikasi menampilkan informasi mendetail tentang data yang dipilih sebelumnya	v	v	v	v	v	v
SIMKES-Fung-03	Kasus Uji *Delete* Data	Aplikasi telah menghapus data tersebut	Aplikasi telah menghapus data tersebut	v	v	v	v	v	v
SIMKES-Fung-04	Kasus Uji *Add* Data	Aplikasi telah menyimpan data tersebut	Aplikasi telah menyimpan data tersebut	v	v	v	v	v	v
SIMKES-Fung-05	Kasus Uji *Edit* Data	Aplikasi telah menyimpan data yang telah dirubah tersebut	Aplikasi telah menyimpan data yang telah dirubah tersebut	v	v	v	v	v	v

SIMKES-Fung-06	Kasus Uji *Maps*	Aplikasi menampilkan *maps* dengan perbedaan warna setiap tingkat persebaran data yang telah difilter sebelumnya	Aplikasi menampilkan *maps* dengan perbedaan warna setiap tingkat persebaran data yang telah difilter sebelumnya	v	v	v	v	v	v
SIMKES-Fung-07	Kasus Uji *Logout*	Aplikasi menampilkan halaman *login* setelah berhasil melakukan *logout.*	Aplikasi menampilkan halaman *login* setelah berhasil melakukan *logout.*	v	v	v	v	v	v

Berdasarkan pada Tabel 7.20 pengujian fungsional yang telah dilakukan, aplikasi mampu untuk melakukan semua kebutuhan fungsional pada enam perangkat yang berbeda dengan baik dan tanpa kendala. Kesimpulan dari hasil dan analisis pengujian fungsional ini bahwa aplikasi memiliki fungsionalitas yang baik untuk perangkat uji yang telah disediakan dengan tingkat keberhasilan sangat baik. Dengan adanya hasil tersebut, diharapkan aplikasi ini memiliki tingkat fungsionalitas yang tinggi pada perangkat lainnya.

7.4 Pegujian *Compatibility*

Pengujian *Compatibility* akan dilakukan dengan menguji aplikasi dengan pemasangan pada enam perangkat berbeda yang telah disebutkan pada sub bab 7.1. Dengan aplikasi operasi dan spesifikasi yang berbeda akan dilakukan beberapa kasus uji dan hasil dari pengujian *compatibility* akan dikatakan valid jika hasil pengujian sesuai dengan implementasi yang telah dilakukan. Pengujian *compatibility* akan dilakukan dengan cara membuat kasus uji dan kemudian akan dilakukan proses pengujian serta akan dibandingkan dengan hasil pengujian (Zhang, et al., March 2015).

7.4.1 Kasus Uji *Compatibility*

Kasus uji akan dijeaskan secara keseluruhan dalam Tabel 7.21 berikut.

Tabel 7.21 Kasus Uji *Compatibility*

Nomor Kasus Uji	Nama Kasus Uji	Hasil yang diharapkan
SIMKES-Comp-01	Kasus Uji *Login*	Aplikasi berhasil menampilkan tampilan halaman *login* yang sesuai dengan implementasi sebelumnya.
SIMKES-Comp-02	Kasus Uji *View* Data	Aplikasi berhasil menampilkan tampilan halaman *view* data yang sesuai dengan implementasi sebelumnya.
SIMKES-Comp-03	Kasus Uji *Delete* Data	Aplikasi berhasil menampilkan tampilan halaman *delete* data yang sesuai dengan implementasi sebelumnya.
SIMKES-Comp-04	Kasus Uji *Add* Data	Aplikasi berhasil menampilkan tampilan halaman *add* data yang sesuai dengan implementasi sebelumnya.
SIMKES-Comp-05	Kasus Uji *Edit* Data	Aplikasi berhasil menampilkan tampilan halaman *edit* data yang sesuai dengan implementasi sebelumnya.
SIMKES-Comp-06	Kasus Uji *Maps*	Aplikasi berhasil menampilkan tampilan halaman *maps* yang sesuai dengan implementasi sebelumnya.

7.4.2 Hasil Analisis Pengujian *Compatibility*

Hasil pengujian *compatibility* akan dijelaskan menggunakan tabel hasil pengujian *compatibility* yang nantinya akan menjadi kesimpulan untuk pengujian *compatibility*. Berikut merupakan penjelasan secara mendetail dari hasil pengujian fungsional akan ditunjukkan pada Tabel 7.22.

Tabel 7.22 Kasus Uji *Compatibility*

Nomor Kasus Uji	Nama Kasus Uji	Hasil yang diharapkan	Hasil Pengujian	Status Perangkat					
				1	2	3	4	5	6
SIMKES-Comp-01	Kasus Uji *Login*	Aplikasi berhasil menampilkan tampilan halaman *login* yang sesuai dengan implementasi sebelumnya.	Aplikasi berhasil menampilkan tampilan halaman *login* yang sesuai dengan implementasi sebelumnya.	v	v	v	v	v	v
SIMKES-Comp-02	Kasus Uji *View* Data	Aplikasi berhasil menampilkan tampilan halaman *view* data yang sesuai dengan implementasi sebelumnya.	Aplikasi berhasil menampilkan tampilan halaman *view* data yang sesuai dengan implementasi sebelumnya.	v	v	v	v	v	v
SIMKES-Comp-03	Kasus Uji *Delete* Data	Aplikasi berhasil menampilkan tampilan halaman *delete* data yang sesuai dengan implementasi sebelumnya.	Aplikasi berhasil menampilkan tampilan halaman *delete* data yang sesuai dengan implementasi sebelumnya.	v	v	v	v	v	v
SIMKES-Comp-04	Kasus Uji *Add* Data	Aplikasi berhasil menampilkan tampilan halaman *add* data yang sesuai dengan implementasi sebelumnya.	Aplikasi berhasil menampilkan tampilan halaman *add* data yang sesuai dengan implementasi sebelumnya.	v	v	v	v	v	v
SIMKES-Comp-05	Kasus Uji *Edit* Data	Aplikasi berhasil menampilkan tampilan halaman *edit* data yang sesuai dengan implementasi sebelumnya.	Aplikasi berhasil menampilkan tampilan halaman *edit* data yang sesuai dengan implementasi sebelumnya.	v	v	v	v	v	v
SIMKES-Comp-06	Kasus Uji *Maps*	Aplikasi berhasil menampilkan tampilan halaman *maps* yang sesuai dengan implementasi sebelumnya.	Aplikasi berhasil menampilkan tampilan halaman *maps* yang sesuai dengan implementasi sebelumnya.	v	v	v	v	v	v

Berdasarkan pada Tabel 7.22 pengujian *compatibility* yang telah dilakukan, aplikasi mampu untuk melakukan menampilkan tampilan yang sesuai dengan implementasi sebelumnya pada enam perangkat yang berbeda dengan baik. Kesimpulan dari hasil dan analisis pengujian *compatibility* ini bahwa aplikasi memiliki *compatibility* yang baik untuk perangkat uji yang telah disediakan dengan tingkat keberhasilan 100%. Dengan adanya hasil tersebut, aplikasi ini memiliki tingkat *compatibility* yang tinggi pada perangkat lainnya.

BAB 8 PENUTUP

8.1 Kesimpulan

Berdasarkan hasil penelitian ini yang dilakukan oleh penulis dengan melakukan analisis kebutuhan, perancangan aplikasi, implementasi serta pengujian aplikasi, maka dapat diambil kesimpulan sebagai berikut:

1. Analisis kebutuhan yang didapatkan berdasarkan kebutuhan pengguna, menghasilkan 13 kebutuhan fungsional, 1 kebutuhan non-fungsional serta 2 jenis pengguna yang diimplementasikan menjadi aplikasi berbasis *native android.*
2. Perancangan aplikasi informasi kesehatan berbasis *mobile native android* terdapat fungsi untuk dapat melakukan pemrosesan data seperti *add, edit, delete,* dan *view*. Perancangan fungsi tersebut kemudian dilakukan dengan membuat arsitektur aplikasi, perancangan *database,* perancangan *interface, activity diagram, sequence diagram,* dan *class diagram.*
3. Implementasi aplikasi dilakukan dalam bentuk aplikasi berbasis *native android* yang menggunakan *platform android studio* serta *google firebase*. Implementasi kode program diimplementasikan menggunakan bahasa pemrograman *java,xml,* serta *javascript* dan memanfaatkan layanan *firebase realtime database,* serta *firebase authenticate* untuk mendukung performa aplikasi.
4. Hasil pengujian aplikasi dilakukan dengan pengujian unit, pengujian fungsional, dan pengujian *compatibility*. Tingkat pengujian fungsional dan pengujian *compatibility* dapat dikategorikan baik karena memiliki prosentase yang tinggi sebesar 100%. Tingkat kompleksitas yang diuji pada pengujian unit juga telah sesuai dengan alur algoritma fitur aplikasi.
5. Penerapan informasi tingkat persebaran data pada *Google Maps* dilakukan dengan implementasi persebaran jumlah data dengan perbedaan warna pada tingkat persebaran data yang diperoleh dari perhitungan prosentase jumlah total dengan jumlah parameter yang telah disediakan pada aplikasi.

8.2 Saran

Saran untuk pengembangan aplikasi selanjutnya adalah sebagai berikut:

1. Fitur untuk *add, edit, view,* dan *delete* untuk data ibu hamil, dan data tubercolosis.
2. Perbaikan *User Interface* dan *User Experience* untuk kenyamanan dan kepuasan pengguna dalam menggunakan aplikasi ini.
3. Penambahan atribut filter pada filter data *maps.*

DAFTAR PUSTAKA

Agusli, R., Sakuroh, L. & Nopriyadi, 2016. Perancangan Sistem Informasi Kesehatan (Puskesmas Keliling) Berbasis Web. *Jurnal Sisfotek Global,* VI(2), pp. 47-53.

Anwar, S. N., Amin, F. & Nugroho, I., 2014. Desain UML Aplikasi Navigasi Layanan Kesehatan Berbasis Android. *Seminar Nasional Sistem Informasi Indonesia,* pp. 250-254.

Damayanti, D. S., Rusmin, M. & Arranury, Z., 2015. Gambaran Penerapan Sistem Informasi Manajemen Kesehatan Berbasis WEB di Puskesmas Kota Makassar Tahun 2015. *Al-Sihah : Public Health Science Journal,* VII(2), pp. 193-202.

Dinas Kesehatan Kota Malang, 2017. *Sistem Informasi Kesehatan Masyarakat,* Malang: Dinas Kesehatan Kota Malang.

Ecma International, 2013. *The JSON Data Interchange Format.* 1st ed. s.l.:Ecma International.

Fowler, M. et al., n.d. *UML Distilled : A Brief Guide to the Standard Object Modelling Language.* 3rd ed. s.l.:Chief Technologist.

Google Inc. & Android Developers, 2017. *Developer Android.* [Online] Available at: *https://developer.android.com/* [Accessed 31 July 2017].

Google, 2018. *Google Firebase.* [Online] Available at: *https://firebase.google.com/* [Accessed 2018].

Haverbeke, M., 2018. *Eloquent JavaScript.* 3rd ed. s.l.:No Starch Press.

IBM Corporation, 2012. *Native, web, or hybrid mobile-app development,* USA: WebSphere.

Isnawati, K., Nugroho, E. & Lazuardi, L., 2016. Implementasi Aplikasi Sistem Informasi Kesehatan Daerah. *Journal of Information Systems for Public Health,* I(1), pp. 64-71.

Kumar S, P., 2014. Analysis of Native and Cross-Platform Methods for Mobile Application Development. *White Paper,* pp. 1-9.

Kurniawati, R., 2016. Pengembangan Sistem Informasi Kependudukan Berbasis Mobile dan RESTFul Web Service. *Seminar Nasional Teknologi Informasi dan Komunikasi 2016 (SENTIKA 2016).*

Meier, R., 2012. *Professional Android™ 4 Application Development.* 1st ed. Indianapolis, Canada: John Wiley & Sons, Inc..

Michael, Ross & Cole, n.d. *Native mobile apps : The wrong choice for business?,* London: michael, ross & cole, ltd. (mrc).

Moroney, L., 2017. *The Definitive Guide to Firebase: Build Android Apps on Google's Mobile Platform.* 1st ed. Seattle, Washington, USA: Apress Media.

Optimus Information, 2015. *Native, Hybrid, or Mobile Web Aplication Development,* Canada: Optimus Information Inc.

Pressman, R. S., 2010. *Software Engineering A Practitioner's Approach.* 7th ed. New York: McGraw-Hill Companies, Inc..

Ramadhan, T. & Utomo, V. G., 2014. Rancang Bangun Aplikasi Mobile untuk Notifikasi Jadwal Kuliah Berbasis Android. *Jurnal Teknologi Informasi dan Komunikasi,* V(2).

Razaq, J. A. & Junanto, A., 2014. Sistem Informasi Publik Layanan Kesehatan menggunakan Metode Location Based Service di Kota Semarang. *Jurnal Teknologi Informasi DINAMIK,* 19(1), pp. 59-67.

Rischpater, R., 2015. *JavaScript JSON Cookbook.* 1st ed. Birmingham, UK: Packt Publishing Ltd..

Rozaq, A., Tolle, H. & Fanani, L., 2018. Pembangunan Aplikasi Brawijaya Messenger dengan menggunakan Platform Firebase pada Universitas Brawijaya. *Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer,* 2(2), pp. 667-673.

Safaat H, N., 2012. *Pemrograman Aplikasi Mobile Smartphone dan Tablet PC Berbasis Android.* Revisi 2 ed. Bandung: Informatika Bandung.

Schildt, H., 2007. *Java™ The Complete Reference.* 7th ed. United State of America: The McGraw-Hill Companies.

Singh, N., 2016. Study of Google Firebase API for Android. *International Journal of Innovative Research in Computer,* 4(9).

Sommerville, I., 2011. *Software Engineering.* 9th ed. New York: Pearson Education, Inc..

Song, I.-Y. & Froehlichh, K., 2013. Entity-relationship modeling. *IEEE Article,* pp. 29-34.

StatCounter, 2017. *StatCounter Global Stats.* [Online] Available at: *http://gs.statcounter.com/* [Accessed 25 July 2018].

Susilowati, S. & Riasti, B. K., 2011. Pembuatan Sistem Informasi Klinik Rawat Inap. 3(1), pp. 1-6.

Tun, P. M., 2014. Choosing a Mobile Application Development Approach. *Asean Journal of Management and Innovation,* I(1), pp. 69-74.

Tutorials Point Ltd., 2016. *Node.js.* 1st ed. s.l.:Tutorials Point Pvt. Ltd..

UU, 2009. *UUI Kesehatan No. 36.* Indonesia, Patent No. 1,2.

Wardiana, W., 2002. *Perkembangan Teknologi Informasi di Indonesia.* Bandung, Wawan Wardiana.

Whitten, J. L. & Bentley, L. D., 2007. *System Analysis & Design Methods.* 7th ed. New York: The McGraw-Hill Companies,Inc..

Wibowo, A., 2018. PERANCANGAN APLIKASI KONSULTASI IBU HAMIL BERBASIS CLOUD COMPUTING. *Perancangan Aplikasi Konsultasi Ibu Hamil berbasis Cloud Computing,* 17(2).

Zhang, T., Gao, J., Cheng, J. & Uehara, T., March 2015. Compatibility Testing Service for Mobile Applications. *2015 IEEE Symposium on Service-Oriented System Engineering*.